



PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: Jeffrey R. Boulter, et al. Examiner: Jerry B. Dennison
Serial No. 09/709,234 Group Art Unit: 2143
Filed: November 9, 2000 Docket No. 85804-019401
Title: INTERNET RADIO AND BROADCAST METHOD
Customer No.: 32361

**DECLARATION UNDER 37 C.F.R. § 1.131 OF
TODD M. BEAUPRE**

Commissioner for Patents
Post Office Box 1450
Alexandria, Virginia 22313-1450

Sir/Madam:

I, Todd M. Beaupre, declare and state that:

1. I am a citizen of the United States.
2. In 1997, I graduated from American University, with a bachelor of science degree in Mathematical Statistics and Economics.
3. I am presently employed by Yahoo! Inc. as Director for Radio/Personalization. I began working for Yahoo! Inc. in 2001, when Yahoo! Inc. acquired Launch Media, Inc., for which I began working in 1999.
4. From 1998 to 1999, I was employed as a Software Engineer with Microsoft. Prior to that, I was employed from 1996 to 1998 as an Engineer with Agents, Inc., which changed its name to Firefly Network in 1996.

BEST AVAILABLE COPY

5. Because of my education and work experience, I am very knowledgeable regarding systems for streaming media content over the Internet.

6. I and Jeffrey R. Boulter (hereinafter collectively referred to as "We") are the named inventors of U.S. Provisional Patent Application No. 60/164,846, entitled "Internet Radio and Broadcast Method", and the present non-provisional patent application which claims the benefit of this provisional application. Both applications are concerned with technology for streaming media content over the Internet.

7. Prior to October 19, 1999, We determined that a user's experience on the Internet could be improved by providing the user with streaming media content, allowing the user to provide feedback, such as a rating, regarding the media content, and selecting media content using received feedback.

8. Accordingly and prior to October 19, 1999, We conceived of a system, which in part, would operate to deliver media content to a user via network, such as the Internet, by selecting the media content for the user influenced using feedback received from the user. The feedback can be in a form of a rating providing by the user, for example.

9. Exhibits 1 to 3 are three Hypertext Markup Language (HTML) documents which provide a description of aspects of the claimed invention, which we conceived of prior to October 19, 1999. These HTML documents are all dated prior to October 19, 1999.

10. The HTML document entitled "LAUNCHcast 2.0: Player UI specification" (Exhibit 1) illustrates and describes, inter alia, an exemplary media player user interface for use with the system, which includes a ratings widget, and buttons (e.g., play, pause, skip, volume, etc.) allowing interactive user feedback.

11. The HTML document entitled “LAUNCHCast 2.0: Media Gateway” (Exhibit 2) describes, inter alia, a gateway program including functionality to stream media content to a user. For example, in response to receipt of a song request, the program checks login credentials, and if the credentials are in order, identifies a song from a play list, locates the file containing the identified song, reads the song file from its location and writes it out to the player.

12. The HTML document entitled “LAUNCHcast 2.0: Playlist Creator” (Exhibit 3) describes, inter alia, selecting songs for a play list. Rating information, as well as other considerations (e.g., the number of songs for a given artist already selected or the number of songs from a given album selected) can be used in selecting songs for a play list.

13. Prior to October 19, 1999, We began writing, or caused to be written, program code implementing a system embodying subject matter of the claimed invention. Exhibit 5 provides a listing of program code, which is in the format of class files written using the Java® Programming Language. Exhibit 4 provides a table of contents of the class files included in Exhibit 5, together with a page number location for each of the class files in Exhibit 5.

14. Included in the program code are “PlaylistGeneratorServlet” (Exhibit 5, pp. 145 to 149) and “PlaylistGenerator” (Exhibit 5, pp. 127 to 144) used to create a play list. More particularly, when the “PlaylistGeneratorServlet” class determines that a new playlist is to be created (See Exhibit 5, p. 146), it calls on the “PlaylistGenerator” and the “create” (Exhibit 5, pp. 139 to 140) program code, which results in the invocation of “createPlaylist” (Exhibit 5, p. 138) to perform various initializations, “gatherMedia” (Exhibit 5, pp. 131 to 133), which gathers the media and other information, including ratings information, from the database, “processSongs” (Exhibit 5, pp. 133 to 137), which identifies scoring information for songs using user ratings

feedback, for example, and assigns songs to various vectors, such as “excluded”, “explicit”, “implicit” and “unrated”, and “makePlaylist” (Exhibit 5, p. 140), which calls “pickSongs” (Exhibit 5, pp. 141 to 143) to pick songs for the playlist using the scoring information and vectors generated by “processSongs”.

15. The “RatingWidgetServlet” code (Exhibit 5, pp. 177 to 183) manages ratings events, including a rating user interface input request and a rating retrieval request for an item such as a song, album or artist, for example. A received rating is saved to a data store, and can be used to update a ratings cache. A request to retrieve a rating can be received, for example, as part of play list generation performed using the “PlaylistGeneratorServlet” class, which class is discussed above.

16. The “MediaGatewayServlet” code (Exhibit 5, pp. 87 to 93) functions, inter alia, to create a relationship between the end user’s media player, the database, and a media server, log media access, and retrieve a user’s play list. Various timestamps can be set for use in determining whether and when to play a song or other media content. A file containing the media to be played can be located, the contents of which can be read for playback by a media player, for example.

17. Prior to October 19, 1999, development of the program code was commenced. For a period commencing prior to October 19, 1999, We diligently pursued development and testing of the program code, which activity yielded the program code attached hereto as Exhibit 5. Exhibit 6 includes electronic mail messages evidencing continuous activity during a period prior to October 19, 1999 through November 5, 1999, which resulted in the program code attached as Exhibit 5.

18. I consider a person of ordinary skill in the art of the claimed invention to be a creative programmer knowledgeable about web development technologies with the ability to solve challenging problems. Such a person would have a strong programming aptitude and either a degree in engineering, preferably electrical or software, or several years of experience developing programs for use with the Internet.

19. Based upon my experience, I believe that a person of ordinary skill in the art would readily understand the disclosure in the HTML documents attached as Exhibits 1 to 3, and the program code attached hereto as Exhibit 5, and recognize them as establishing conclusively (1) that the claimed invention was conceived of prior to October 19, 1999, and (2) that the claimed invention will operate satisfactorily for its intended purpose.

20. Having made this discovery, on or before October 28, 1999, We promptly engaged counsel for the purposes of preparing and filing a patent application on October 28, 1999, as evidenced by the electronic mail messages dated October 28 and 29, 1999 (Exhibit 6, page 71), and the facsimile cover sheet dated October 29, 1999 (Exhibit, page 72).

21. Between October 28, 1999 and November 4, 1999, We exchanged communications with Andrew S. Jordan, of the law firm of Cislo & Thomas. (Exhibit 6, pages 83 and 86).

22. The provisional patent application was filed on November 10, 1999. The provisional patent application was assigned provisional no. 60/164,846 (hereinafter referred to as "Application No. '846"). (Exhibit 7) The provisional patent application is the parent of the present non-provisional application, the priority of the '846 having been claimed in the present application. (Exhibit 8)

23. In support of this Declaration, the copied documents presented in Exhibits 1 to 8 were all collected by my attorneys at Greenberg Traurig, LLP.

24. I declare further that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

Date: 12/7/06

By: Todd M. Beaupre
Todd M. Beaupre

LAUNCHcast 2.0: Player UI specification

Discuss this page on the LaunchCast Bulletin Board

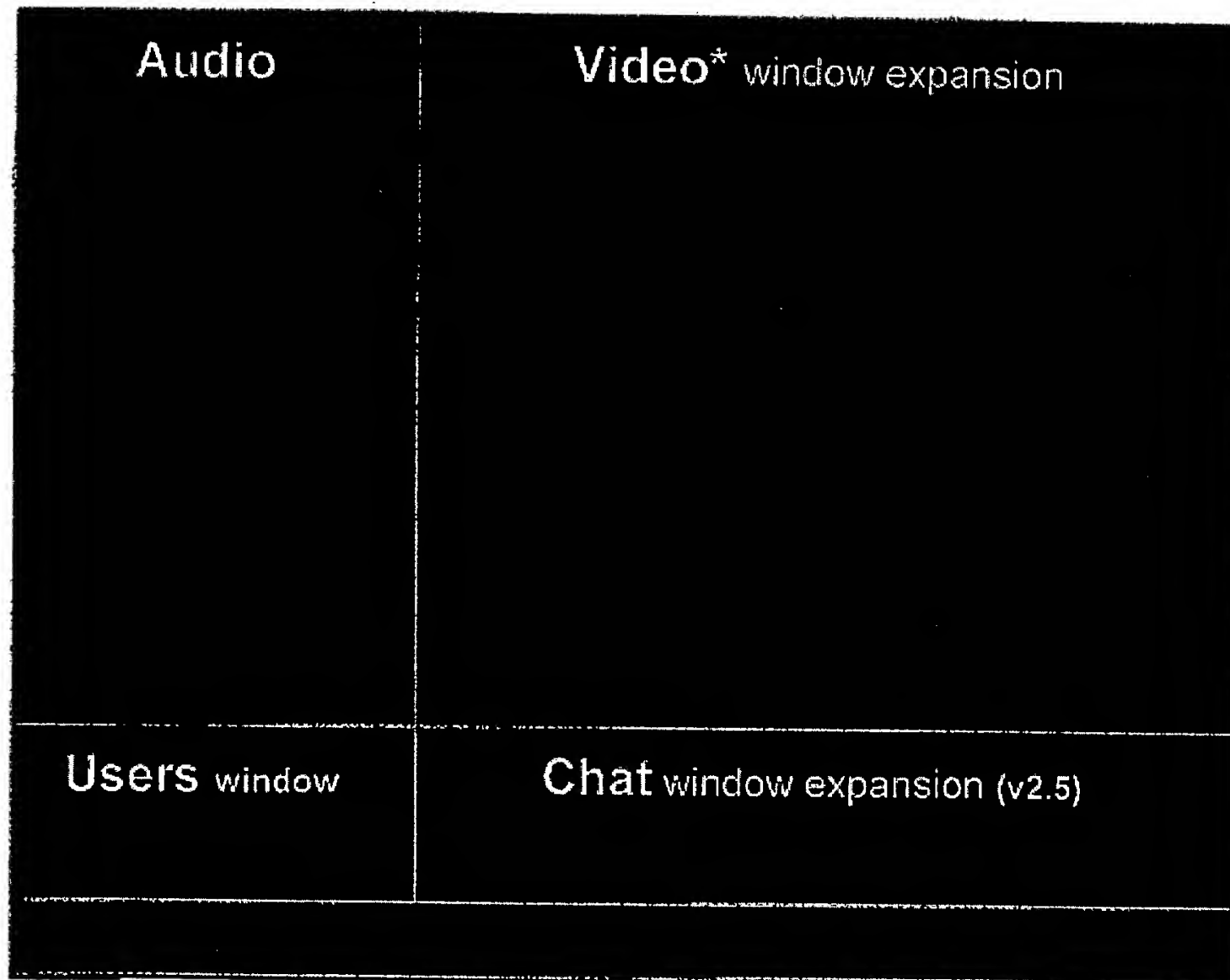
DRAFT

Size: small enough that the user can keep it open while using other applications

Browser: no client download. supports browsers consistent with rest of LAUNCH.com (3.0+)

The LAUNCHcast player window will contain the following features:

Expandable window for video display and chat, maybe something like:



Play/pause button

Skip song (active only on non-community LAUNCHcasts)

Volume control

Prominent rating widget

- Shows current explicit rating

- Shows implicit rating (and source) when explicit one not present

- Shows other songs with same rating value (when user puts mouse over a rating value?)

- Graphically shows community popularity (average rating) and ranking, where appropriate

- Gives rewards (factoids?) when user rates item

Current song title (linked to song page)

Artist (linked to artist page)

Album (linked to album page)

Link to fans of the song (ordered by rating and online status)

Area to display text tips and factoids

Small image advertisement

[VIDEO >] button appears active when video available... expands window and starts video, otherwise links to video section

[CHAT >] button allows user to open and close chat interface on community LAUNCHcasts

button appears active when digital download available... downloads in separate window, otherwise links to downloads section

[RATE MORE] button (linked to list of more songs to rate)

[BUY] button (linked to album/single commerce page)

[OPTIONS] button (linked to LAUNCHcast options)

[HELP] button (linked to player tutorial)

All links open in the same target browser window unless otherwise specified.

LAUNCHcast 2.0 | Last Updated: 5/17/99 2:13 PM

LaunchCast 2.0: Media Gateway

Discuss this page on the LaunchCast Bulletin Board

DRAFT

Overview

The Media Gateway is accessed via HTTP and used to play a song in LaunchCast 2.0. It performs a number of tasks and if all criteria are met, streams out a media file (audio or video).

This program can be written in TCL for StoryServer or could be a compiled program in Java or C++ if performance requires.

Tasks

1. Check cookies for valid Launch login credentials. In the case of an error, play an audio clip that asks the user to log in
2. Checks the USER_AGENT HTTP header to make sure a user isn't trying to download a file with a browser. If the check fails, redirect to <http://www.launch.com>
3. Write out the HTTP headers including an expires header and MIME type appropriate for the media
4. Open the file for reading in binary mode
5. Select the top 1 song from in the the playlist by ordinal in the playlist from the database. Delete the row in the database. Look up its file name and path. If either of these last two tasks fails, exit.
6. Call the stored procedure OnPlayStart. Close the database connection. Ignore errors.
7. Read the file from disk and write it out as raw data to the client. In the case of an error, exit.
8. When finished, call the stored procedure OnPlayEnd. Ignore errors.

LaunchCast 2.0 | Last Updated: 5/11/99 4:28 PM

LAUNCHcast 2.0: Playlist Creator

Discuss this page on the LaunchCast Bulletin Board

DRAFT

Retrieve user's player preferences (including audio/video options)

Get ratings data

Ratings will be scaled from the album and artist 1-7 scale to a 0-100 scale for songs. This is independent of the interface; we may still show 1-7 in the song rating widget. This gives us a lot of flexibility for changing the rating scale in the future however. The mapping from the old scale to the new is as follows:

Old Rating	Old Rating Text	New Rating
7	The Best	90
6	Great	75
5	I like that	60
4	It's OK	40
3	Not my thing	30
2	Pretty Bad	20
1	Hate It	0

The higher the rating, the more often a song will be played. A rating of 0 means the song will never be played again for a particular user.

If this is the first time a user has used LaunchCast 2.0, their album and artist ratings are propagated down to song ratings. When LaunchCast 2.0 is released, stored procedures will be added to the back end of both the album and song rating widgets so all those ratings are propagated to song ratings.

For video playlists, the follow queries will select among only those songs with videos available.

Retrieve user's explicit song ratings

Retrieve user's implicit album->song ratings and populate implicit rating matrix, where we don't have an explicit song rating

Retrieve user's implicit artist->song ratings and populate rating matrix, where we don't have an explicit song rating or album->song rating

Retrieve average rating from user's DJs for all songs rated by their advisors and populate rating matrix

Retrieve and scale user's implicit BDS playlist->song ratings and populate rating matrix

Retrieve and scale user's venue->artists->song ratings and populate rating matrix (note: we will retrieve a maximum of 50 artists across all venues, so retrieve 50/venue memberships artists from each venue.) If a song is present in any venue, it receives a score of 100 for the implicit venue rating

Retrieve and scale user's song recommendations from NetPerceptions and populate rating matrix

Retrieve community average rating for all songs in the rating matrix and add to matrix

Retrieve last played and scale time since played logarithmically on a scale of 1 to 100.

Multiply rating matrix by weight matrix to compute score

Weight Matrix	Album Exists	Artist Exists	Neither Exists
Album->song rating	30	0	0
Artist->song rating	0	20	0
MyDJs avg rating	15	15	20
BDS Playlist->song rating	15	15	20
Venue->artist->song rating	05	05	05
NetP recommendation rating	05	05	10
Community average rating	05	05	05
Last Played	25	25	20
Total	1.0	0.9	0.8

NOTE: When MyDJs average doesn't exist, community average is used as MyDJs average

Example:

User profile matrix for HitsMan

Song	Album->Song	Artist->Song	MyDJs Avg	BDS Playlist	Venue	NetP	Community Avg	Last Played
Dizz Knee Land	0	0	58.4	0	0	56.9	51.2	56.2
Building A Mystery	75	0	(55.1)	0	100	35.1	35.1	23.1

Army	0	55	78.9	50	100	47.3	52.8	100.0
------	---	----	------	----	-----	------	------	-------

Weight matrix:

Song	Album->Song	Artist->Song	MyDJs Avg	BDS Playlist	Venue	NetP	Community Avg	Last Played
Dizz Knee Land	0	0	0.20	0.20	0.05	0.05	0.05	0.20
Building A Mystery	0.3	0	0.15	0.15	0.05	0.05	0.05	0.25
Army	0	0.2	0.15	0.15	0.05	0.05	0.05	0.25

Score matrix:

Song	Album->Song	Artist->Song	MyDJs Avg	BDS Playlist	Venue	NetP	Community Avg	Last P
Dizz Knee Land	0	0	13.88	0	0	2.85	3.06	11.24
Building A Mystery	22.5	0	8.63	0	5	3.26	2.76	5.78
Army	0	13.0	11.84	9	5	2.37	2.64	25.0

Sort explicit ratings by rating (descending).

Sort implicit ratings by score (descending). So in this example, 'Army' would be most likely to be played since it has the highest score, followed by 'Building a Mystery' and 'Dizz Knee Land'

While $\text{songCount} < \text{playlistLength}$, choose songs biased towards top of both lists, based on user's unrated quota proportions

The determination of which songs are put into the playlist is done with the following formula, with:

f = user determined integer between 1 and 100. The higher the exponent, the more songs are chosen from the top of the list.

ls = number of songs in list

r = random number between 0 and 1 (generated each time)

array index of song to pick = $(r^f) / (ls^f) * ls$

Enforce legal rules (may be different for video):

If the artist of the song chosen has less than 4 songs in this playlist and there are less than 3 songs from this album in the playlist, remove the song from the list add it to the playlist. Otherwise, remove this song from the list and choose again.

Shuffle songs

Write playlist to playlists table

Return ASX file with entries as an infinite repeat of the file streaming URL with the userId and playlistId as a parameters (e.g.

<http://stream.launch.com/stream.xxx?userId=51066&playlistId=3487>)

LAUNCHcast 2.0 | *Last Updated: 5/14/99 11:06 AM*

<u>File Name</u>	<u>Page No.</u>
AlbumArtistData.java	1
AlbumInfo.java	2
ArtistInfo.java	4
AverageRating.java	5
Bandwidth.java	7
BDSRank.java	10
CachedRating.java	11
ClipCollection.java	12
ClipSchedule.java	13
Clip.java	16
Constants.java	21
DBConnection.java	23
DBException.java	27
DBPreparedStatement.java	28
DBResultSet.java	29
DJList.java	33
DJ.java	35
FrequencyCounter.java	36
GeneratorParameters.java	39
GenreIndex.java	41
GenreList.java	43
GetAds.java	45
GetBDSSStations.java	47
GetGenres.java	48
GetItemRatingsFromDB.java	49
GetLastPlayed.java	51
GetNews.java	53
GetPlaylist.java	55
GetPlaylistServers.java	57
GetPlaylistServersInterface.java	59
GetPopular.java	60
GetRatings.java	62
GetRatingsCacheUsers.java	66
GetRatingsCacheUsersInterface.java	69
GetRecentlyPlayed.java	70
GetSongInfoServlet.java	72

<u>File Name</u>	<u>Page No.</u>
GetSongRatingsFromDB.java	80
IntHash.java	81
ItemsProfile.java	82
Item.java	84
MediaFormat.java	86
MediaGatewayServlet.java	87
MediaList.java	94
Media.java	96
PickCount.java	97
PickList.java	100
PickStatus.java	102
PlayDataHash.java	103
PlayDates.java	104
Playlist.java	113
Playlist2.java	123
PlaylistCreatorTest.java	125
PlaylistEntry.java	126
PlaylistGenerator.java	127
PlaylistGeneratorServlet.java	145
PlaylistMaker.java	150
PlaylistParameters.java	151
PlaylistStatus.java	153
PopularSongs.java	156
Population.java	158
Rating.java	167
RatingsCache.java	168
RatingsProfile.java	175
RatingWidgetServlet.java	177
RecList.java	184
SaveClips.java	188
SavePlaylist.java	190
SimpleClip.java	192
SimpleClipList.java	193
SimplePlaylist.java	194
SongData.java	197
SongGroup.java	207
SongInfo.java	208

<u>File Name</u>	<u>Page No.</u>
SongInfoCache.java	211
SongInfoCacheUpdater.java	220
SongList.java	222
SongRating.java	225
Song.java	226
Station.java	228
StationList.java	229
Util.java	230
WeightMatrix.java	233

```

package com.launch.PlaylistGenerator;

public class AlbumArtistData
{
    Item album = null;
    Item artist = null;

    boolean alreadyTriedAlbum = false;
    boolean alreadyTriedArtist = false;

    public void reset()
    {
        album = null;
        artist = null;
        alreadyTriedAlbum = false;
        alreadyTriedArtist = false;
    }

    public Item getAlbum(ItemsProfile items, SongData data)
    {
        if (alreadyTriedAlbum)
            return album;

        alreadyTriedAlbum = true;

        album = items.get(data.getAlbumID());

        return album;
    }

    public Item getArtist(ItemsProfile items, SongData data)
    {
        if (alreadyTriedArtist)
            return artist;

        alreadyTriedArtist = true;

        artist = items.get(data.getArtistID());

        return artist;
    }
}

```

D:\My Documents\email\Launch\PlaylistGenerator\AlbumArtistData.java Page 1 of 1 11/05/99 1:32 PM

```

package com.launch.PlaylistGenerator;

import java.util.Vector;

public class AlbumInfo
{
    int ID;
    String title;
    ArtistInfo artist;

    Vector genres;

    public AlbumInfo(int ID)
    {
        this.ID = ID;
    }

    public String toString()
    {
        return "[albumID=" + ID + ", title=" + title
            + ", genres=" + genresString() + ", artist=" + artist.toString()
+ "]" ;
    }

    public String genresString()
    {
        if (genres == null)
            return "(NONE)";

        String result = "";

        for (int i = 0; i < genres.size(); i++)
        {
            result = result.concat(genres.elementAt(i) + ", ");
        }

        return "(" + result + ")";
    }

    public int getArtistID() throws Exception
    {
        if (artist == null)
            throw new Exception("artist is not set for album " + ID + " (" +
title + ") ");

        return artist.ID;
    }

    public boolean inGenres(short genreID)
    {
        if (genres == null)
            return false;
    }
}

```

```

        return genres.contains(new Short(genreID));
    }

    public boolean inGenres(GenreList userGenres)
    {
        if (userGenres.allGenres == true)
            return true;

        if (genres == null)
            return false;

        // do it the other way, check each of the genres the song is
        // in and if it's in the user's genres

        for (int i = 0; i < genres.size(); i++)
        {
            Short genreID = (Short) genres.elementAt(i);

            if (userGenres.exists(genreID))
                return true;
        }

        return false;
    }

    public void addGenre(short genreID)
    {
        if (genres == null)
            genres = new Vector(1,1);

        // be careful not to add duplicates
        Short genre = new Short(genreID);

        if (!genres.contains(genre))
            genres.addElement(new Short(genreID));
    }
}

```

```

package com.launch.PlaylistGenerator;

import java.util.Hashtable;

public class ArtistInfo
{
    int ID;
    String title;
    Hashtable songs;

    public ArtistInfo(int ID)
    {
        this.ID = ID;
        songs = new Hashtable();
    }

    public String toString()
    {
        return "[artistID=" + ID + ", title=" + title + "]";
    }

    public final static boolean isVariousArtists(int itemID)
    {
        return (itemID == Constants.ARTIST_VARIOUS_ARTISTS
            || itemID == Constants.ARTIST_ORIGINAL_SOUNDTRACK
            || itemID == Constants.ARTIST_SOUNDTRACK);
    }
}

```

D:\My Documents\email\Launch\PlaylistGenerator\ArtistInfo.java Page 1 of 1 11/05/99 1:37 PM


```

package com.launch.PlaylistGenerator;

public class AverageRating extends Rating
{
    private short count = 0;
    private int sum;

    private boolean calculated = false;

    public AverageRating()
    {
        super();
    }

    public AverageRating(short defaultRating)
    {
        super(defaultRating);
    }

    public void add(int value)
    {
        sum += value;
        count++;
        calculated = false;
    }

    public short get()
    {
        calculate();

        return super.get();
    }

    public short count()
    {
        return count;
    }

    private void calculate()
    {
        if (!calculated)
        {
            if (count > 0)
            {
                set(Util.average(count, sum));
                set = true;
            }

            calculated = true;
        }
    }

    public String toString()
    {

```

```
String ratingStr = "(Not calculated)";  
if (set) ratingStr = "" + rating;  
return sum + "/" + count + "=" + ratingStr;
```

```
}
```

```
}  
D:\My Documents\email\Launch\PlaylistGenerator\AverageRating.java Page 2 of 2 11/05/99 1:27 PM
```

```

package com.launch.PlaylistGenerator;

public class Bandwidth
{
    public final static short SPEED_28  = 28;
    public final static short SPEED_56  = 56;
    public final static short SPEED_100 = 100;
    public final static short SPEED_128 = 128;
    public final static short SPEED_300 = 300;
    public final static short SPEED_500 = 500;

    private boolean beenset = false;
    private short value = SPEED_28;

    public Bandwidth()
    {
    }

    public Bandwidth(short speed)
    {
        value = speed;
        beenset = true;
    }

    public Bandwidth(String speed)
    {
        if (speed == null)
        {
            beenset = false;
        }
        else
        {
            if (speed.equals("28"))
                set(SPEED_28);
            else if (speed.equals("56"))
                set(SPEED_56);
            else if (speed.equals("100"))
                set(SPEED_100);
            else if (speed.equals("128"))
                set(SPEED_128);
            else if (speed.equals("300"))
                set(SPEED_300);
            else if (speed.equals("500"))
                set(SPEED_500);
            else
            {
                beenset = false;
            }
        }
    }

    public String toString()
    {

```

```

        if (value == SPEED_28)
            return "28.8k";
        else if (value == SPEED_56)
            return "56k";
        else if (value == SPEED_100)
            return "100k";
        else if (value == SPEED_128)
            return "128k";
        else if (value == SPEED_300)
            return "300k";
        else if (value == SPEED_500)
            return "56k";

        return "UNKNOWN (" + value + ")";
    }

    public short get()
    {
        return value;
    }

    public void set(short speed)
    {
        if (speed == SPEED_28
            || speed == SPEED_56
            || speed == SPEED_100
            || speed == SPEED_128
            || speed == SPEED_300
            || speed == SPEED_500)
        {
            value = speed;
            beenset = true;
        }
        else
            beenset = false;
    }

    public boolean load(DBConnection conn, int userID)
    {
        try
        {
            DBResultSet rs = conn.executeQuery("exec
sp_al50UserPreference_GetValue_xsxx " + userID);

            if (!rs.getBOF() && !rs.getEOF())
            {
                set(rs.getShort("iDefaultBandwidth"));
            }
        }
        catch (DBException oops)
        {
            Util.debug("DB Exception in Bandwidth::load: " +

```

```
oops.getMessage());  
    }
```

```
        return isSet();
```

```
    }
```

```
    public boolean isSet()
```

```
    {
```

```
        return beenset;
```

```
    }
```

```
}
```

D:\My Documents\email\Launch\PlaylistGenerator\Bandwidth.java

Page 3 of 3

11/05/99 1:32 PM

```
package com.launch.PlaylistGenerator;
```

```
public class BDSRank  
{
```

```
    short stationID;  
    byte rank;
```

```
    public BDSRank(short stationID, byte rank)
```

```
    {  
        this.stationID = stationID;  
        this.rank = rank;  
    }
```

```
    public String toString()
```

```
    {  
        return stationID + ":" + rank;  
    }
```

```
}
```

D:\My Documents\email\Launch\PlaylistGenerator\BDSRank.java Page 1 of 1 11/05/99 1:26 PM

```

package com.launch.PlaylistGenerator;

import java.io.*;
import java.util.Date;

/**
 * This class is used to model a single rating in the cache.
 */

public final class CachedRating implements Serializable
{
    public int userID;
    public int itemID;
    public byte rating;
    public byte type;

    private Date created = new Date();

    //-----

    public CachedRating(int userID, int itemID, byte rating, byte type)
    {
        this.userID = userID;
        this.itemID = itemID;
        this.rating = rating;
        this.type = type;
    }

    public final String toString()
    {
        return("user:" + userID + ", itemID:" + itemID + ", rating:" +
rating + ", type:" + typeString(type) + ", date:" + created.toString() +
Util.newLine);
    }

    public final static String typeString(byte type)
    {
        if (type == Constants.ITEM_TYPE_SONG)
            return "song";
        else if (type == Constants.ITEM_TYPE_ALBUM)
            return "album";
        else if (type == Constants.ITEM_TYPE_ARTIST)
            return "artist";

        return "unknown";
    }

    public String hashKey()
    {
        return itemID + ":" + type;
    }
}

```

D:\My Documents\email\Launch\PlaylistGenerator\CachedRating.java Page 1 of 1 11/05/99 1:35 PM

```
package com.launch.PlaylistGenerator;
```

```
import java.util.Hashtable;
```

```
public class ClipCollection extends Hashtable  
{
```

```
    public Clip put(int clipID, Clip aClip)
```

```
    {  
        return (Clip) put(new Integer(clipID), aClip);  
    }
```

```
    public Clip get (int clipID)
```

```
    {  
        return (Clip) get(new Integer(clipID));  
    }
```

```
}
```

D:\My Documents\email\Launch\PlaylistGenerator\ClipCollection.java Page 1 of 1 11/05/99 1:26 PM


```

package com.launch.PlaylistGenerator;

import java.util.Date;
import javax.servlet.ServletOutputStream;

public class ClipSchedule
{
    private Date dbDate;

    private int userID, lastBroadcast, currentBroadcast;

    private boolean set = false;

    public SimplePlaylist playlist;

    public ClipSchedule (int userID)
    {
        this.userID = userID;
    }

    public void init(DBConnection conn)
    {
        set = false;

        try
        {
            DBResultSet rs = conn.executeQuery("exec sp_lcGetClipSchedule_xxxx
" + userID);

            if (!rs.getBOF() && !rs.getEOF())
            {
                dbDate          = rs.getTimestamp("dbDate");
                lastBroadcast    = rs.getInt("lastBroadcastID");
                currentBroadcast = rs.getInt("broadcastID");
                playlist         =
SimplePlaylist.fromBytes(rs.getBytes("playlist"));
            }
            else
            {
                dbDate = new Date();
            }

            // the first time a playlist is created for a user, the dates
will be null

            if (playlist != null)
            {
                if (playlist.lastAd == null) playlist.lastAd = dbDate;
                if (playlist.lastNews == null) playlist.lastNews = dbDate;
                if (playlist.lastTip == null) playlist.lastTip = dbDate;
                set = true;
            }
        }
    }
}

```

```

    }
    catch (DBException e)
    {
        System.err.println("DBException in ClipSchedule::init:" +
e.toString());
    }
}

private long dateDiff(Date diffMe)
{
    if (diffMe == null)
        diffMe = new Date(0);

    return (long) ((dbDate.getTime() - diffMe.getTime()) / (1000.0 * 60));
}

public byte nextClipType(boolean debug, ServletOutputStream out)
{
    long adDiff, newsDiff, tipDiff;

    while (true)
    {
        adDiff = dateDiff(playlist.lastAd);
        newsDiff = dateDiff(playlist.lastNews);
        tipDiff = dateDiff(playlist.lastTip);

        if (debug)
        {
            Util.out(out, "dbDate is " + dbDate.toString());

            Util.out(out, "lastAdDate is " + playlist.lastAd);
            Util.out(out, "next ad in " + (Constants.AD_THRESHOLD -
adDiff) + " minutes");

            Util.out(out, "lastNewsDate is " + playlist.lastNews);
            Util.out(out, "next news clip in " +
(Constants.NEWS_THRESHOLD - newsDiff) + " minutes");

            Util.out(out, "lastTipDate is " + playlist.lastTip);
            Util.out(out, "next tip in " + (Constants.TIP_THRESHOLD -
tipDiff) + " minutes");
        }

        if (playlist == null)
        {
            System.err.println(new Date().toString() + " nextClipType:
userID " + userID + " has no/invalid playlist");
            return Clip.TYPE_NONE;
        }

        if (currentBroadcast > lastBroadcast)
        {
            if (debug) Util.out(out, "getting broadcast");

```

```

        lastBroadcast = currentBroadcast;
        return Clip.TYPE_BROADCAST;
    }
    else if (adDiff >= Constants.AD_THRESHOLD)
    {
        if (debug) Util.out(out, "playing AD");
        playlist.lastAd = dbDate;

        if (playlist.ads.isEmpty())
            System.err.println(new Date().toString() + " userID "
+ userID + " is out of ads");
        else
            return Clip.TYPE_AD;
    }
    else if (newsDiff >= Constants.NEWS_THRESHOLD)
    {
        if (debug) Util.out(out, "playing NEWS");
        playlist.lastNews = dbDate;

        if (playlist.news.isEmpty())
            System.err.println(new Date().toString() + " userID "
+ userID + " is out of news");
        else
            return Clip.TYPE_NEWS;
    }
    else if (tipDiff >= Constants.TIP_THRESHOLD)
    {
        if (debug) Util.out(out, "playing TIP");
        playlist.lastTip = dbDate;

        if (playlist.tips.isEmpty())
            System.err.println(new Date().toString() + " userID "
+ userID + " is out of tips");
        else
            return Clip.TYPE_TIP;
    }
    else
    {
        if (debug) Util.out(out, "playing SONG");

        if (playlist.songs.isEmpty())
        {
            System.err.println(new Date().toString() + " userID "
+ userID + " is out of songs");
            return Clip.TYPE_NONE;
        }
        else
            return Clip.TYPE_SONG;
    }
}
//return Clip.TYPE_NONE;
}

```

```

package com.launch.PlaylistGenerator;

import java.util.Date;

public class Clip
{
    public final static byte TYPE_NONE = 0;
    public final static byte TYPE_NEWS = 1;
    public final static byte TYPE_AD = 2;
    public final static byte TYPE_INTERSTITIAL = 3;
    public final static byte TYPE_TIP = 4;
    public final static byte TYPE_SONG = 5;
    public final static byte TYPE_BROADCAST = 6;

    public int ID;
    public byte type;
    public int mediaID;
    public Date lastPlayed;
    public String name, directory, server, filepath;
    public MediaList media;
    byte origin;

    private boolean set = false;

    public Clip(byte type)
    {
        this.type = type;
        media = new MediaList();
    }

    public Clip(int ID, byte type)
    {
        this(type);
        this.ID = ID;
    }

    public Clip(int ID, byte type, int mediaID, String name, Date lastPlayed)
    {
        this(ID, type);
        this.ID = ID;
        this.mediaID = mediaID;
        this.name = name;
        this.lastPlayed = lastPlayed;
    }

    public byte type() { return type; }

    public boolean isSet() { return set; }

    private void setDirectory(String newDir)
    {
        if (!newDir.equals(" "))
        {
            directory = newDir;
        }
    }
}

```

```

    }

    public void logPlay(DBConnection conn, int userID)
    {
        String sql = "";

        if (type == TYPE_SONG)
            sql = "exec sp_lcLogPlaySong_isud " + userID + ", " +
mediaID + ", " + ID + ", " + origin;
        else if (type == TYPE_AD)
            sql = "exec sp_lcLogPlayAd_isud " + userID + ", " +
mediaID + ", " + ID;
        else if (type == TYPE_NEWS)
            sql = "exec sp_lcLogPlayNews_isud " + userID + ", " +
mediaID + ", " + ID;
        else if (type == TYPE_TIP)
            sql = "exec sp_lcLogPlayTip_isud " + userID + ", " +
mediaID + ", " + ID;
        // else if (type == TYPE_BROADCAST)
        // sql = "exec sp_lcLogPlayBroadcast_isux " + userID + ", " +
mediaType;

        try
        {
            conn.executeUpdate(sql, true);
        }
        catch (DBException e)
        {
            System.err.println("DBException in Clip:logPlay:" +
e.toString());
        }
    }

    public boolean getPath(DBConnection conn, ClipSchedule schedule)
    {
        if (type == TYPE_NONE)
            return false;

        SimpleClipList list = null;

        if (type == TYPE_SONG)
            list = schedule.playlist.songs;
        else if (type == TYPE_AD)
            list = schedule.playlist.ads;
        else if (type == TYPE_TIP)
            list = schedule.playlist.tips;
        else if (type == TYPE_NEWS)
            list = schedule.playlist.news;

        if (list == null)
            return false;

        SimpleClip yip = list.pop();
    }

```

```

        if (yip == null)
            return false;

        mediaID = yip.mediaID;
        ID = yip.ID;
        origin = yip.origin;

        try
        {
            DBResultSet rs = conn.executeSQL("exec sp_lcGetMediaPath_xxxx " +
mediaID);

            if (!rs.getBOF() && !rs.getEOF())
            {
                setDirectory(rs.getString("directory"));
                server = rs.getString("server");
                filepath = rs.getString("filepath");

                set = true;
            }
        }
        catch (DBException e)
        {
            System.err.println("DBException in Clip::getPath: " +
e.toString());
        }

        return set;
    }

    /*
    public boolean pop(DBConnection conn, int userID, int context)
    {
        set = false;

        try
        {
            DBResultSet rs;
            String the_command;

            int contextNum = 0;
            if (context > 1) contextNum = 1;

            if (type==TYPE_BROADCAST)
            {
                the_command="exec " + BROADCAST_SP + " " + userID + ", " +
type + ", " + context;
            }
            else
            {
                String stored_proc = null;

                if (type == TYPE_AD ) stored_proc = ADS_SP;

```

```

        else if (type == TYPE_TIP ) stored_proc = TIPS_SP;
        else if (type == TYPE_NEWS) stored_proc = NEWS_SP;
        else
            stored_proc = SONG_SP;

        the_command= "exec " + stored_proc + " " + userID + ", " +
contextNum;
    }

    rs = conn.executeQuery(the_command);

    if (!rs.getBOF() && !rs.getEOF())
    {
        setDirectory(rs.getString("directory"));
        server      = rs.getString("server");
        filepath    = rs.getString("filepath");

        set = true;
    }
}
catch (DBException e)
{
    System.err.println("DBException in Clip::pop: " + e.toString());
}

return isSet();
}
*/

public String path()
{
    return server
        + directory
        + "/"
        + filepath;
}

public String toString()
{
    return "Clip type (" + typeName() + "), id = " + mediaID
        + ", lastPlayed = " + lastPlayed
        + ", media = " + media.toString()
        + ", path = " + path();
}

public PlaylistEntry toPlaylistEntry(short mediaType)
{
    PlaylistEntry entry = new PlaylistEntry();
    entry.mediaID = media.getID(mediaType);
    entry.title   = name;

    entry.filepath = media.getFilepath(mediaType);
}

```

```

        return entry;
    }

    public SimpleClip toSimpleClip(short mediaType)
    {
        return new SimpleClip(ID, media.getID(mediaType));
    }

    public String typeName()
    {
        switch(type)
        {
            case TYPE_AD:
                return "Ad";
            case TYPE_BROADCAST:
                return "Broadcast";
            case TYPE_INTERSTITIAL:
                return "Interstitial";
            case TYPE_NEWS:
                return "News";
            case TYPE_TIP:
                return "Tip";
            case TYPE_SONG:
                return "Song";
        }

        return "?";
    }

    public String URL()
    {
        return server
            + directory
            + "/"
            + filepath;
    }
}

```

D:\My Documents\email\Launch\PlaylistGenerator\Clip.java Page 5 of 5 11/05/99 1:32 PM


```

package com.launch.PlaylistGenerator;

public interface Constants
{
    // live

    /*
    public final static String DB_SOURCE                = "LAUNCHcast";
    public final static String DB_USERNAME              = "dbClient";
    public final static String DB_PASSWORD              = "83kareem23";
    public final static String DB_DBNAME                = "dbLaunchProd";
    public final static String DB_SERVER                = "209.67.158.19"; //
DB3
    public final static short DB_PORT                  = 1433;

    public final static String STREAM_URL =
"http://lcplaylist.launch.com/servlet/gateway";
    public final static String STREAM_SERVER = "http://lcstream.launch.com";
    */

    // development

    public final static String DB_SOURCE                = "LAUNCHcast";
    public final static String DB_USERNAME              = "dbClient";
    public final static String DB_PASSWORD              = "29Idiocy99";
    public final static String DB_DBNAME                = "dbLaunchProd";
    public final static String DB_SERVER                = "zeus";
    public final static short DB_PORT                  = 1433;

    public final static String STREAM_URL =
"http://devweb7.launch.com/servlet/gateway";
    public final static String STREAM_SERVER = "http://devweb7.launch.com/F";

    public final static int RIAA_MAX_SONGS_FROM_ALBUM    = 2;
    public final static int RIAA_MAX_SONGS_BY_ARTIST     = 3;

    public final static int BDS_SCORE_MAX_POINTS        = 41;
    public final static int BDS_SCORE_POINTBAR          = 20;

    public final static int DEFAULT_LASTPLAYED_SCORE    = 100;
    public final static int DEFAULT_MEDIATYPE           = 211; // 16
Mono
    public final static int DEFAULT_UNRATED_RATIO       = 50;
    public final static int DEFAULT_PICK_FACTOR         = 7;
    public final static int DEFAULT_BDS_SCORE           = 0;
    public final static int MAX_PERCENT_RATED_SONGS_TO_PICK = 20;
    public final static int NEW_USER_UNRATED_RATIO      = 90;
    public final static int MIN_RATINGS_TO_HONOR_RATIO  = 100;
    public final static int MIN_SIZE_FOR_NO_UNRATED     = 200;
    public final static int MAX_ORDINAL                  = 1000;

    // for calculating implicit based on other song ratings
    public final static int MAX_SONGS_BY_ARTIST         = 4;

```

```

// random picking
public final static int RANDOM_SONGS_COUNT = 5000;
// this is a percent of the total number of songs in the database
public final static int MIN_SONGS_IN_GENRES_TO_GET_RANDOM = 5;

public final static int MIN_RATING_FOR_RATED_SOURCE = 35;
// songs with average rating above this are considered popular
// also change this at the top of LAUNCHCast/player/getsonginfo
public final static int POPULAR_THRESHOLD = 58;

public final static int DEFAULT_RATING = 52; //
global average for all songs
public final static int DEFAULT_DJS_SCORE = DEFAULT_RATING;
public final static int DEFAULT_NETP_SCORE =
DEFAULT_RATING;
public final static byte DEFAULT_COMMRATING =
DEFAULT_RATING;

public final static int MAX_RATINGS_TO_GET = 500;
public final static int MAX_DJ_RATINGS_TO_GET = 500;

public final static int ARTIST_VARIOUS_ARTISTS = 1028125;
public final static int ARTIST_ORIGINAL_SOUNDTRACK = 1020156;
public final static int ARTIST_SOUNDTRACK = 1036715;
public final static int DEFAULT_PLAYLIST_SIZE = 50;
public final static int MAX_NEWS_ITEMS = 0;
public final static int MAX_ADS = 20;
public final static int MAX_TIPS_ITEMS = 0;

public final static int REFRESH_AT_SONGS_LEFT = 8;
public final static int REFRESH_AT_NEW_RATINGS_COUNT = 15;

public final static int AD_THRESHOLD = 30;
public final static int NEWS_THRESHOLD = 99999999;
public final static int TIP_THRESHOLD = 99999999;

public final static byte ITEM_TYPE_SONG = 1;
public final static byte ITEM_TYPE_ALBUM = 2;
public final static byte ITEM_TYPE_ARTIST = 3;

// the size of the ratings cache FOR EACH user
public final static int RATINGS_CACHE_INITIAL_SIZE = 2000;

public final static int RATING_UPDATE_LIST_INITIAL_SIZE = 100;

// for updating the ratings caches
public static final int PROPAGATE_DIRTY_RATING_SLEEP_TIME = 60 * 1000; //
every 60 seconds

public static final String POST_HEADER = "POST /servlet/playlist HTTP/1.0";

public static final int PORT_NUMBER = 80;

```

```

package com.launch.PlaylistGenerator;

import java.util.Properties;

import com.inet.tds.TdsDriver;

import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Connection;
import java.sql.Driver;
import java.sql.DriverManager;
import java.util.Date;

public class DBConnection
{
    private Connection conn;

    public static Driver DBDriver;

    public DBConnection() throws DBException
    {
        if (DBConnection.DBDriver == null)
            DBConnection.initializeDriver();

        if (DBConnection.DBDriver == null)
            return;

        String url = "jdbc:inetdae:"
            + Constants.DB_SERVER
            + ":"
            + Constants.DB_PORT
            + "?sql7=true&database="
            + Constants.DB_DBNAME
            + "&user="
            + Constants.DB_USERNAME
            + "&password="
            + Constants.DB_PASSWORD
            + "";

        try
        {
            conn = DBConnection.DBDriver.connect(url, null);
        }
        catch (SQLException oops)
        {
            throw new DBException(oops);
        }
        catch (Exception err)
        {
            Util.debug("Exception: " + err.toString());
        }
    }

    private static void initializeDriver()

```

```

    {
        DBDriver = new com.inet.tds.TdsDriver();
    }

    private DBResultSet execute(String sql, boolean printSQL) throws DBException
    {
        if (printSQL)
            Util.debug(Util.newLine + Thread.currentThread().getName() + "
Running SQL: " + sql);

        DBResultSet myRs = new DBResultSet();

        try
        {
            // if we don't have a query, don't run it. It'll hang
            if (sql.length() <= 0)
                return myRs;

            Statement query = conn.createStatement();

            if (query.execute(sql))
            {
                myRs.setResultSet(query.getResultSet());
            }
        }
        catch (SQLException oops)
        {
            System.err.println(Util.newLine + (new Date()).toString() + "
DBException: " + Thread.currentThread().getName() + " Running SQL: " + sql + ",
exception: " + oops.toString());
            oops.printStackTrace();
            throw new DBException(oops);
        }

        return myRs;
    }

    public void executeUpdate(String sql, boolean printSQL) throws DBException
    {
        if (printSQL)
            Util.debug(Util.newLine + Thread.currentThread().getName() + "
Running SQL: " + sql);

        try
        {
            // if we don't have a query, don't run it. It'll hang
            if (sql.length() <= 0)
                return;

            Statement query = conn.createStatement();

```

```

        query.executeUpdate(sql);
    }
    catch (SQLException oops)
    {
        // when we call a stored proc that gets a text pointer this
happens,
        // so ignore it
        if (oops.getMessage().indexOf("Unknown datatype") > -1)
        {
            //          System.err.println("ignoring unknown datatype exception");
            return;
        }

        System.err.println(Util.newLine + (new Date()).toString() + "
DBException: " + Thread.currentThread().getName() + " Running SQL: " + sql + ",
exception: " + oops.toString());
        oops.printStackTrace();
        throw new DBException(oops);
    }
}

public DBResultSet executeSQL(String sql) throws DBException
{
    return execute(sql, true);
}

public DBResultSet executeSQL(String sql, boolean printSQL) throws
DBException
{
    return execute(sql, printSQL);
}

public DBPreparedStatement prepareStatement(String sql) throws DBException
{
    try
    {
        return new DBPreparedStatement(conn.prepareStatement(sql));
    }
    catch (SQLException oops)
    {
        System.err.println(Util.newLine + (new Date()).toString() + "
DBException in prepareStatement: " + Thread.currentThread().getName() + ",
exception: " + oops.toString());
        oops.printStackTrace();
        throw new DBException(oops);
    }
}

public boolean close() throws DBException

```

```

    {
        if (conn == null)
            return false;

        try
        {
            conn.close();
            conn = null;
            return true;
        }
        catch (SQLException oops)
        {
            throw new DBException(oops);
        }
    }

    public void finalize() throws DBException
    {
        // in case someone forgets
        close();
    }
}

```

D:\My Documents\email\Launch\PlaylistGenerator\DBConnection.java Page 4 of 4 11/05/99 1:37 PM

```
package com.launch.PlaylistGenerator;

import java.sql.SQLException;

public class DBException extends Exception
{
    SQLException oops;

    public DBException(SQLException oops)
    {
        this.oops = oops;
    }

    public String getMessage()
    {
        return oops.toString();
    }
}
```

D:\My Documents\email\Launch\PlaylistGenerator\DBException.java

Page 1 of 1

11/05/99 1:26 PM

```

package com.launch.PlaylistGenerator;

import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Date;

public class DBPreparedStatement
{
    PreparedStatement statement;

    public DBPreparedStatement(PreparedStatement statement)
    {
        this.statement = statement;
    }

    public void setBytes(int parameterIndex, byte x[]) throws DBException
    {
        try
        {
            if (statement != null)
            {
                statement.setBytes(parameterIndex, x);
            }
        }
        catch (SQLException e)
        {
            throw new DBException(e);
        }
    }

    public void executeUpdate() throws DBException
    {
        Util.debug(Util.newLine + Thread.currentThread().getName() + " Running
prepared statement");

        if (statement == null)
            return;

        try
        {
            statement.executeUpdate();
        }
        catch (SQLException oops)
        {
            System.err.println(Util.newLine + (new Date()).toString() + "
DBException: " + Thread.currentThread().getName() + " Running Statement, exception:
" + oops.toString());
            oops.printStackTrace();
            throw new DBException(oops);
        }
    }
}

```

D:\My Documents\email\Launch\PlaylistGenerator\DBPreparedStatement.java Page 1 of 1 11/05/99 1:32 PM


```

package com.launch.PlaylistGenerator;

import java.util.Date;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Timestamp;
import java.io.InputStream;

public class DBResultSet
{
    private ResultSet rs;
    private boolean atEOF = false;
    private boolean atBOF = true;

    public void setResultSet(ResultSet aRS) throws DBException
    {
        try
        {
            rs = aRS;

            if (rs != null)
                atBOF = !rs.next();
        }
        catch (SQLException oops)
        {
            throw new DBException(oops);
        }
    }

    public int getInt(String columnName) throws DBException
    {
        try
        {
            return rs.getInt(columnName);
        }
        catch (SQLException oops)
        {
            throw new DBException(oops);
        }
    }

    public int getInt(int position) throws DBException
    {
        try
        {
            return rs.getInt(position);
        }
        catch (SQLException oops)
        {
            throw new DBException(oops);
        }
    }

    public InputStream getAsciiStream(String columnName) throws DBException

```

```

    {
        try
        {
            return rs.getAsciiStream(columnName);
        }
        catch (SQLException oops)
        {
            throw new DBException(oops);
        }
    }

    public short getShort(String columnName) throws DBException
    {
        try
        {
            return rs.getShort(columnName);
        }
        catch (SQLException oops)
        {
            throw new DBException(oops);
        }
    }

    public boolean getBoolean(String columnName) throws DBException
    {
        try
        {
            return rs.getBoolean(columnName);
        }
        catch (SQLException oops)
        {
            throw new DBException(oops);
        }
    }

    public byte[] getBytes(String columnName) throws DBException
    {
        try
        {
            return rs.getBytes(columnName);
        }
        catch (SQLException oops)
        {
            throw new DBException(oops);
        }
    }

    public float getFloat(String columnName) throws DBException
    {
        try
        {
            return rs.getFloat(columnName);
        }
    }

```

```

    }
    catch (SQLException oops)
    {
        throw new DBException(oops);
    }
}

public float getFloat(int position) throws DBException
{
    try
    {
        return rs.getFloat(position);
    }
    catch (SQLException oops)
    {
        throw new DBException(oops);
    }
}

public String getString(String columnName) throws DBException
{
    try
    {
        return rs.getString(columnName);
    }
    catch (SQLException oops)
    {
        throw new DBException(oops);
    }
}

public Date getDate(String columnName) throws DBException
{
    try
    {
        return rs.getDate(columnName);
    }
    catch (SQLException oops)
    {
        throw new DBException(oops);
    }
}

public Timestamp getTimestamp(String columnName) throws DBException
{
    try
    {
        return rs.getTimestamp(columnName);
    }
    catch (SQLException oops)
    {
        throw new DBException(oops);
    }
}

```

```

public boolean getBOF() throws DBException
{
    return atBOF;
}

public boolean getEOF() throws DBException
{
    return atEOF;
}

public void next() throws DBException
{
    try
    {
        atEOF = !rs.next();
    }
    catch (SQLException oops)
    {
        throw new DBException(oops);
    }
}

public boolean wasNull() throws DBException
{
    try
    {
        return rs.wasNull();
    }
    catch (SQLException oops)
    {
        throw new DBException(oops);
    }
}
}

```

D:\My Documents\email\Launch\PlaylistGenerator\DBResultSet.java Page 4 of 4 11/05/99 1:32 PM

```

package com.launch.PlaylistGenerator;

import java.util.Vector;

public class DJList extends Vector
{
    public DJ djAt(int i)
    {
        return (DJ) elementAt(i);
    }

    public String inList()
    {
        Integer list[] = new Integer[size()];

        int last = 0;

        for (int i = 0; i < this.size(); i++)
        {
            list[i] = new Integer(djAt(i).userID);
        }

        return Util.join(", ", list);
    }

    public boolean load(DBConnection conn, int userID, int moodID)
    {
        short djCount = 0;

        try
        {
            DBResultSet rs = conn.executeQuery("exec sp_lcoGetDJs_xxxx "
                                                + userID + ", "
                                                + moodID);

            while (!rs.getBOF() && !rs.getEOF())
            {
                addElement(new DJ(rs.getInt("djID")));

                rs.next();
                djCount++;
            }

            Util.debug(Thread.currentThread().getName() + " added " + djCount
+ " DJs");
        }
        catch (DBException oops)
        {
            Util.debug("DB Exception in DJList::load: " + oops.getMessage());
        }
    }
}

```

```

        return (djCount > 0);
    }

    public Vector asIDVector()
    {
        Vector users = new Vector(10);

        for (int i = 0; i < this.size(); i++)
        {
            users.addElement(new Integer(((DJ) elementAt(i)).userID));
        }

        return users;
    }
}

```

D:\My Documents\email\Launch\PlaylistGenerator\DJList.java Page 2 of 2 11/05/99 1:28 PM

```
package com.launch.PlaylistGenerator;
```

```
public class DJ
```

```
{
```

```
    public int userID;
```

```
    public String alias;
```

```
    public DJ (int id, String name)
```

```
    {
```

```
        this(id);
```

```
        alias = name;
```

```
    }
```

```
    public DJ (int id)
```

```
    {
```

```
        userID = id;
```

```
    }
```

```
}
```

```
D:\My Documents\email\Launch\PlaylistGenerator\DJ.java
```

```
Page 1 of 1
```

```
11/05/99 1:26 PM
```

```

package com.launch.PlaylistGenerator;

import java.util.*;

/**
 * FrequencyCounter is a Hashtable of the form (Object, Integer)
 * <br><br>
 * okay I realize the getLargest and getSmallestValue
 * methods are very inefficient (CPU wise) but these methods
 * aren't called often, if they are then some one should
 * do an nlog(n) sort on them then just pick out the largest
 * after that
 */
public class FrequencyCounter extends Hashtable
{
    public FrequencyCounter()
    {
    }

    public FrequencyCounter(int i)
    {
        super(i);
    }

    public void incrementValue(Object o)
    {
        Integer i=(Integer)get(o);

        if (i==null)
        {
            put(o, new Integer(1));
        }
        else
        {
            put(o, new Integer((i.intValue()+1)));
        }
    }

    public FrequencyCounter getLargest(int n)
    {
        FrequencyCounter fc=new FrequencyCounter(n+10);

        Integer temp_int;
        Object temp_object;

        Object smallest_value_key=null;
        int smallest_value;

        Enumeration e=keys();

        while (e.hasMoreElements())
        {

```



```

        temp_object=e.nextElement();
        temp_int=(Integer)get(temp_object);

        if (fc.size()>=n)
        {
            smallest_value_key=fc.getSmallestValue();

smallest_value=((Integer)fc.get(smallest_value_key)).intValue();

            if (temp_int.intValue()>smallest_value)
            {
                fc.remove(smallest_value_key);
                fc.put(temp_object, temp_int);
            }
        }
        else
        {
            fc.put(temp_object, temp_int);
        }
    }

    return(fc);
}

```

```

/** @return null if list is empty */
public Object getSmallestValue()
{
    int smallest_value=Integer.MAX_VALUE;
    Object smallest_value_key=null;

    int temp_int;
    Object temp_object;

    Enumeration e=keys();
    while(e.hasMoreElements())
    {
        temp_object=e.nextElement();
        temp_int=((Integer)get(temp_object)).intValue();

        if (temp_int<smallest_value)
        {
            smallest_value=temp_int;
            smallest_value_key=temp_object;
        }
    }

    return(smallest_value_key);
}

```

```

//*****
// The following is a test function

public static void main(String argv[])

```

```

    {
        FrequencyCounter fc=new FrequencyCounter();

        fc.incrementValue("one");

        fc.incrementValue("two");
        fc.incrementValue("two");

        fc.incrementValue("three");
        fc.incrementValue("three");
        fc.incrementValue("three");

        fc.incrementValue("four");
        fc.incrementValue("four");
        fc.incrementValue("four");
        fc.incrementValue("four");

        System.out.println(fc);
        System.out.println("smallest "+ fc.getSmallestValue());
        System.out.println("largest 2" + fc.getLargest(2));
    }
}
D:\My Documents\email\Launch\PlaylistGenerator\FrequencyCounter.java      Page 3 of 3      11/05/99 1:28
PM

```

```

package com.launch.PlaylistGenerator;

import javax.servlet.http.HttpServletRequest;

public class GeneratorParameters
{
    private int userID, moodID, djID;
    private Bandwidth speed;
    private boolean debug, matrix, forceRefresh, dontsave;
    private MediaFormat format;

    private boolean moodIDSet    = false;
    private boolean djIDSet      = false;

    private int debugFormat = Util.DISPLAY_TEXT;

    public Bandwidth speed()
    {
        return speed;
    }

    public MediaFormat format()
    {
        return format;
    }

    public int debugFormat()
    {
        return debugFormat;
    }

    public int userID()
    {
        return userID;
    }

    public int moodID()
    {
        return moodID;
    }

    public int djID()
    {
        if (djIDSet)
            return djID;

        return userID;
    }

    public boolean debug()
    {
        return debug;
    }
}

```

```

public boolean matrix()
{
    return matrix;
}

public boolean forceRefresh()
{
    return forceRefresh;
}

public boolean dontsave()
{
    return dontsave;
}

public GeneratorParameters(HttpServletRequest request)
{
    debug          = (request.getParameter("ralph")          != null);
    matrix          = (request.getParameter("matrix")        != null);
    forceRefresh    = (request.getParameter("forceRefresh")  != null);
    dontsave        = (request.getParameter("dontsave")      != null);

    String debugFormatString = request.getParameter("format");

    if (debugFormatString != null && debugFormatString.equals("html"))
        debugFormat = Util.DISPLAY_HTML;

    try { userID = Integer.parseInt(request.getParameter("u")); }
    catch (NumberFormatException e) { userID = 0; }

    try { moodID = Integer.parseInt(request.getParameter("m")); }
    catch (NumberFormatException e) { moodID = 0; moodIDSet = false; }

    moodIDSet = true;

    try { djID = Integer.parseInt(request.getParameter("d")); }
    catch (NumberFormatException e) { djID = userID; djIDSet = false; }

    djIDSet = true;

    if (djID <= 0)
    {
        djID = userID;
        djIDSet = false;
    }

    speed = new Bandwidth(request.getParameter("b"));

    format = new MediaFormat();
}
}

```

D:\My Documents\email\Launch\PlaylistGenerator\GeneratorParameters.java Page 2 of 2 11/05/99 1:24 PM

```

package com.launch.PlaylistGenerator;

import java.util.Hashtable;
import java.util.Vector;

public class GenreIndex extends Hashtable
{
    public GenreIndex(int x, int y)
    {
        super(x, y);
    }

    public void add(short index, SongInfo info)
    {
        SongList list = get(index);

        if (list == null)
        {
            list = new SongList();
            put(new Short(index), list);
        }

        list.addElement(info);
    }

    public SongList get(int index)
    {
        return (SongList) get(new Short((short) index));
    }

    public int countInGenreList(GenreList myGenres)
    {
        int result = 0;

        SongList list;

        for (int i = 0; i < myGenres.size(); i++)
        {
            list = get(myGenres.genreAt(i));

            if (list != null)
            {
                result += list.size();
            }
        }

        return result;
    }

    /**
     * returns a COPY of the list of songs in genres
     */
    public SongList getInGenreList(GenreList myGenres)

```

```

    {
        SongList result = new SongList();

        for (int i = 0; i < myGenres.size(); i++)
        {
            result.addElements(get(myGenres.genreAt(i)));
        }

        return result;
    }

    /**
     * returns a COPY of the list of songs in a genre
     */
    public SongList getInGenre(int genreID)
    {
        SongList list = get(genreID);
        SongList result;

        if (list == null)
            list = new SongList();

        result = (SongList) list.clone();

        return result;
    }
}

```

}
 D:\My Documents\email\Launch\PlaylistGenerator\GenreIndex.java

Page 2 of 2 11/05/99 1:28 PM

```

package com.launch.PlaylistGenerator;

import java.util.Hashtable;

public class GenreList
{
    private int genres[];
    private Hashtable hash;

    private byte next;

    public boolean allGenres = true;

    public GenreList()
    {
        hash = new Hashtable(1,1);
        genres = new int[100];
    }

    public int add(short genreID)
    {
        allGenres = false;
        hash.put(new Short(genreID), new Boolean(true));
        genres[next] = genreID;
        next++;

        return genres[next - 1];
    }

    public int size()
    {
        return next;
    }

    public int genreAt(int pos)
    {
        return genres[pos];
    }

    public boolean exists(Short genreID)
    {
        if (next == 0)
            return true;

        else
            return hash.containsKey(genreID);
    }

    public String toString() {

        String result = "";

        for (int i = 0; i < size(); i++)
        {

```

```
        result = result.concat(genreAt(i) + ", ");  
    }  
    return result;  
}  
}
```

D:\My Documents\email\Launch\PlaylistGenerator\GenreList.java Page 2 of 2 11/05/99 1:26 PM


```

package com.launch.PlaylistGenerator;

import java.util.Date;
import java.util.Vector;

public class GetAds extends Thread
{
    Vector ads;
    int userID;
    short mediaType;

    public GetAds(Vector ads, int userID, short mediaType)
    {
        this.ads      = ads;
        this.userID    = userID;
        this.mediaType = mediaType;
    }

    public void run()
    {
        Date startDate = new Date();
        Thread.currentThread().setName("GetAds");

        int rowCount = 0;
        int count     = 0;

        Clip aClip;
        int clipID, mediaID;
        Date lastPlayed;
        String clipName;

        String sql = new String("exec sp_lcGetAds_xsxx "
                                + userID
                                + ", "
                                + mediaType
                                );

        try
        {
            DBConnection conn = new DBConnection();
            DBResultSet rs = conn.executeQuery(sql);

            while (!rs.getBOF() && !rs.getEOF() && count < Constants.MAX_ADS)
            {
                ads.addElement(new Clip(rs.getInt("clipID"),
                                       Clip.TYPE_AD,
                                       rs.getInt("mediaID"),
                                       rs.getString("clipName"),
                                       rs.getDate("lastPlayed")));

                count++;
                rs.next();
                rowCount++;
            }
        }
    }
}

```

```

        conn.close();
    }
    catch (DBException oops)
    {
        Util.debug("DB Exception: " + oops.getMessage());
    }

    Util.debug(Thread.currentThread().getName() + " added " + count + "
ads");
    Util.printElapsedTime(Thread.currentThread().getName(), startDate);
}

```

```

}
D:\My Documents\email\Launch\PlaylistGenerator\GetAds.java Page 2 of 2 11/05/99 1:37 PM

```

```

package com.launch.PlaylistGenerator;

import java.util.Date;

public class GetBDSStations extends Thread
{
    int userID;
    int moodID;
    StationList stations;

    public GetBDSStations(int userID, int moodID, StationList stations)
    {
        this.userID = userID;
        this.moodID = moodID;
        this.stations = stations;
    }

    public void run()
    {
        Date startDate = new Date();
        Thread.currentThread().setName("GetBDSStations");

        int rowCount = 0;

        String sql = "sp_lcGetBDSNames_xsxx " + userID + ", " + moodID;

        try
        {
            DBConnection conn = new DBConnection();

            DBResultSet rs = conn.executeSQL(sql);

            while (!rs.getBOF() && !rs.getEOF())
            {
                int bdsID = rs.getInt("bdsID");
                stations.addElement(new Station(bdsID));
                rowCount++;
                rs.next();
            }

            conn.close();
        }
        catch (DBException oops)
        {
            Util.debug("DB Exception in GetBDSStations: " +
oops.getMessage());
        }

        Util.debug(Thread.currentThread().getName() + " got " + rowCount + "
BDS station subscriptions");
        Util.printElapsedTime(Thread.currentThread().getName(), startDate);
    }
}

```

```

package com.launch.PlaylistGenerator;

import java.util.Date;

public class GetGenres extends Thread
{
    GenreList genres;
    int djID;
    int moodID;

    public GetGenres(GenreList genres, int djID, int moodID)
    {
        this.genres = genres;
        this.moodID = moodID;
        this.djID = djID;
    }

    public void run()
    {
        Date startDate = new Date();
        Thread.currentThread().setName("GetGenres");

        int rowCount = 0;

        try
        {
            DBConnection conn = new DBConnection();

            DBResultSet rs = conn.executeQuery("exec
sp_lcGetGenreNamesForUser_xsxx "
                                                + djID + ", "
                                                + moodID);

            while (!rs.getBOF() && !rs.getEOF())
            {
                genres.add((short) rs.getInt("genreID"));
                rowCount++;
                rs.next();
            }

            conn.close();
        }
        catch (DBException oops)
        {
            Util.debug("DB Exception: " + oops.getMessage());
        }

        Util.debug(Thread.currentThread().getName() + " added " + rowCount + "
genres");
        Util.printElapsedTime(Thread.currentThread().getName(), startDate);
    }
}

```

```

package com.launch.PlaylistGenerator;

import java.util.*;

public final class GetItemRatingsFromDB extends Thread
{
    private Vector userIDs;
    private Vector results;

    //-----

    public GetItemRatingsFromDB(Vector userIDs, Vector results)
    {
        this.userIDs = userIDs;
        this.results = results;
    }

    public void run()
    {
        Thread.currentThread().setName("GetItemRatingsFromDB");
        Util.debug(Thread.currentThread().getName() + " thread started");
        Date startDate = new Date();

        try
        {
            String sql = "SELECT iUserID_FK userID, iSourceTableID_L
type, iItemID_FK itemID, tiRating rating FROM a125ItemRating WHERE iUserID_FK IN ("
+ RatingsCache.GetVectorAsCommaDelimitedList(userIDs) + ')';

            DBConnection conn = new DBConnection();
            DBResultSet rs = conn.executeSQL(sql);
            CachedRating cr;

            byte type;

            while (!rs.getBOF() && !rs.getEOF())
            {
                cr = new CachedRating(rs.getInt("userID"),
rs.getInt("itemID"), (byte) rs.getInt("rating"),
sourceTableIDToType(rs.getInt("type")));
                results.addElement(cr);
                rs.next();
            }

            conn.close();
        }
        catch (DBException oops)
        {
            System.err.println("DBException in GetItemRatingsFromDB: "
+ oops.getMessage());
        }

        Util.printElapsedTime(Thread.currentThread().getName(),
startDate);
    }
}

```

```
public final static byte sourceTableIDToType (int type)
{
    if (type == 260)
        return Constants.ITEM_TYPE_ARTIST;

    // assume album (243)

    return Constants.ITEM_TYPE_ALBUM;
}
```

```
}
D:\My Documents\email\Launch\PlaylistGenerator\GetItemRatingsFromDB.java Page 2 of 2 11/05/99 1:32
PM
```

```

package com.launch.PlaylistGenerator;

import java.util.Date;
import java.text.DateFormat;
import javax.servlet.ServletOutputStream;

public class GetLastPlayed extends Thread
{
    PlayDates lastPlayed;
    int userID;
    ServletOutputStream out;

    public GetLastPlayed(PlayDates lastPlayed, int userID, ServletOutputStream
out)
    {
        this.lastPlayed = lastPlayed;
        this.userID      = userID;
        this.out          = out;
    }

    public void run()
    {
        Date startDate = new Date();
        Thread.currentThread().setName("GetLastPlayed");

        // returns: songID, lastPlayed

        try
        {
            DBConnection conn = new DBConnection();

            Util.printElapsedTime(Thread.currentThread().getName() + " got a
dbConnection", startDate);

            lastPlayed.load(conn, userID);

            Util.printElapsedTime(Thread.currentThread().getName() + " loaded
dates", startDate);

            // this is somewhat expensive, so only do it every so often
            if (Util.random(10) == 1)
            {
                Util.debug("resaving lastPlayed for user " + userID);
                lastPlayed.save(conn);
            }

            conn.close();
        }
        catch (DBException oops)
        {
            Util.debug("DB Exception: " + oops.getMessage());
        }

        Util.out(out, Thread.currentThread().getName() + " loaded " +

```

```
lastPlayed.size() + " dates");  
    Util.printElapsedTime(Thread.currentThread().getName() + "done  
GetLastPlayed", startDate);  
}
```

```
}  
D:\My Documents\email\Launch\PlaylistGenerator\GetLastPlayed.java Page 2 of 2 11/05/99 1:35 PM
```



```

package com.launch.PlaylistGenerator;

import java.util.Date;
import java.util.Vector;

public class GetNews extends Thread
{
    Vector news;
    int userID;
    short mediaType;
    int moodID;

    public GetNews(Vector news, int userID, short mediaType, int moodID)
    {
        this.news = news;
        this.userID = userID;
        this.mediaType = mediaType;
        this.moodID = moodID;
    }

    public void run()
    {
        Date startDate = new Date();
        Thread.currentThread().setName("GetNews");

        int rowCount = 0;
        int count = 0;

        Clip aClip;
        int clipID, mediaID;
        Date lastPlayed;
        String clipName;

        /*
        sp_lcGetNews_xsxx      @userID int, @moodID int, @mediaType int
        returns clipID, clipName, mediaID, lastPlayed
        */

        String sql = new String("exec sp_lcGetNews_xsxx "
                                + userID
                                + ", "
                                + moodID
                                + ", "
                                + mediaType
                                );

        try
        {
            DBConnection conn = new DBConnection();
            DBResultSet rs = conn.executeQuery(sql);

            while(!rs.getBOF() && !rs.getEOF() && count <
Constants.MAX_NEWS_ITEMS)

```

```

        {
            news.addElement(new Clip(rs.getInt("clipID"),
                                    Clip.TYPE_NEWS,
                                    rs.getInt("mediaID"),
                                    rs.getString("clipName"),
                                    rs.getDate("lastPlayed")));

            count++;
            rs.next();
            rowCount++;
        }

        conn.close();
    }
    catch (DBException oops)
    {
        Util.debug("DB Exception: " + oops.getMessage());
    }

    Util.debug(Thread.currentThread().getName() + " added " + count + "
news items");
    Util.printElapsedTime(Thread.currentThread().getName(), startDate);
}

}
D:\My Documents\email\Launch\PlaylistGenerator\GetNews.java Page 2 of 2    11/05/99 1:38 PM

```

```

package com.launch.PlaylistGenerator;

import java.util.Date;

public class GetPlaylist extends Thread
{
    Population songs;
    int userID;
    SongInfoCache cache;

    public GetPlaylist(Population songs, int userID, SongInfoCache cache)
    {
        this.songs = songs;
        this.userID = userID;
        this.cache = cache;
    }

    public void run()
    {
        Date startDate = new Date();
        Thread.currentThread().setName("GetPlaylist");

        SongInfo info = null;
        SimpleClip clip;
        int songID;
        int rowCount = 0;

        try
        {
            DBConnection conn = new DBConnection();
            Util.printElapsedTime(Thread.currentThread().getName() + " got a
dbConnection", startDate);

            SimplePlaylist playlist = SimplePlaylist.load(conn, userID);

            if (playlist != null)
            {
                for (int i = 0; i < playlist.songs.size(); i++)
                {
                    clip = (SimpleClip) playlist.songs.elementAt(i);
                    songID = clip.ID;

                    songs.initSong(songID, Song.EXCLUDED);
                    info = (SongInfo) cache.get(songID,
SongInfoCache.TYPE_SONG);

                    songs.artistCounts.increment(info.album.artist.ID);
                    songs.albumCounts.increment(info.album.ID);

                    rowCount++;
                }
            }

            conn.close();
        }
    }
}

```

```
        catch (DBException oops)
        {
            Util.debug("DB Exception: " + oops.getMessage());
        }

        Util.debug(Thread.currentThread().getName() + " excluded " + rowCount +
" songs");
        Util.printElapsedTime(Thread.currentThread().getName(), startDate);
    }
}
```

D:\My Documents\email\Launch\PlaylistGenerator\GetPlaylist.java Page 2 of 2 11/05/99 1:34 PM

```

package com.launch.PlaylistGenerator;

import java.util.*;

/**
 **/

public final class GetPlaylistServers extends Thread
{
    public static int SLEEP_TIME = (3600*1000); // every hour

    public static int EXPECTED_SERVER_COUNT = 10;

    private GetPlaylistServersInterface personToNotify;

    //-----

    /**
     * @param personToNotify must not be null.
     **/
    public GetPlaylistServers(GetPlaylistServersInterface personToNotify)
    {
        this.personToNotify=personToNotify;
    }

    public void run()
    {
        Thread.currentThread().setName("getPlaylistServers");
        Util.debug(Thread.currentThread().getName() + " thread started");
        DBConnection conn;
        DBResultSet rs;
        Vector v;
        Date benchmark_date;

        try
        {
            while (personToNotify!=null)
            {
                benchmark_date=new Date();
                v=new Vector(EXPECTED_SERVER_COUNT);
                conn = new DBConnection();
                rs = conn.executeQuery("exec
sp_lcGetRatingsCacheServers_xsxd");

                while (!rs.getBOF() && !rs.getEOF())
                {
                    v.addElement(rs.getString("server"));
                    rs.next();
                }

                conn.close();
                personToNotify.updatePlaylistServers(v);

                Util.printElapsedTime(Thread.currentThread().getName() + ", get " + v.size())

```

```

+ " rows", benchmark_date);

        Thread.sleep(SLEEP_TIME);
    }
    catch (Exception e)
    {
        System.err.println(new Date().toString() + " Fatal
Exception in GetPlaylistServers:" + e.toString());
    }

    Util.debug(Thread.currentThread().getName() + " thread done");
}
}

```

D:\My Documents\email\Launch\PlaylistGenerator\GetPlaylistServers.java
PM

Page 2 of 2 11/05/99 1:37

```
package com.launch.PlaylistGenerator;

import java.util.*;

public interface GetPlaylistServersInterface
{
    /**
     * @param playlistServers will be a vector of strings, each string is an ip
     address of the form xxx.xxx.xxx.xxx
     */
    public void updatePlaylistServers(Vector playlistServers);
}
D:\My Documents\email\Launch\PlaylistGenerator\GetPlaylistServersInterface.java    Page 1 of 1
11/05/99 1:28 PM
```

```

package com.launch.PlaylistGenerator;

import java.util.Date;

public class GetPopular extends Thread
{
    Population songs;
    SongList list;

    public GetPopular(Population songs, SongList list)
    {
        this.songs    = songs;
        this.list      = list;
    }

    public void run()
    {
        Date startDate = new Date();
        Thread.currentThread().setName("GetPopular");
        Song ditty;
        SongData data;
        SongInfo info;

        int rowCount = 0;

        if (list != null)
        {
            for (int i = 0; i < list.size(); i++)
            {
                info = list.elementAt(i);

                data = songs.getSongData(info.songID);

                if (data != null)
                {
                    // we can't add it, but let's append the info while
                    data.setInfo(info);
                }
                else
                {
                    data = songs.initSongGetData(info.songID,
Song.UNRATED);

                    if (data != null)
                    {
                        data.querySource = data.SOURCE_POPULAR;
                        data.setInfo(info);
                    }
                    rowCount++;
                }
            }
        }
    }
}

```



```
    }  
    Util.debug(Thread.currentThread().getName() + " added " + rowCount + "  
songs");  
    Util.printElapsedTime(Thread.currentThread().getName(), startDate);  
}
```

```
}  
D:\My Documents\email\Launch\PlaylistGenerator\GetPopular.java    Page 2 of 2    11/05/99 1:38 PM
```

```

package com.launch.PlaylistGenerator;

import java.util.Date;
import java.util.Vector;
import java.util.Enumeration;
import javax.servlet.ServletOutputStream;

public class GetRatings extends Thread
{
    ItemsProfile profile;
    int userID;
    DJList djs;
    Population songs;
    SongInfoCache cache;
    ServletOutputStream out;

    → public GetRatings(Population songs, ItemsProfile profile, int userID, DJList
        djs, SongInfoCache cache, ServletOutputStream out)
    {
        this.profile = profile;
        this.userID = userID;
        this.djs = djs;
        this.cache = cache;
        this.songs = songs;
    }

    public void run()
    {
        Date startDate = new Date();
        Thread.currentThread().setName("GetRatings");

        int rowCount = 0;

        // make a users vector from the users and djs

        Vector users = djs.asIDVector();
        users.addElement(new Integer(userID));

        Util.out(out, "GetRatings getting ratings for users " +
users.toString());

        Vector ratings = cache.ratingsCache.getRatings(users);

        Util.printElapsedTime("GetRatings after all ratings retrieved",
startDate);

        CachedRating cached;
        int djID, itemID;
        byte rating, type;
        SongData data;
        short songType = Song.EXPLICIT;
        SongInfo info;
        int artistID;
        Item theItem;
    }
}

```

```

int songRatings = 0;
int itemRatings = 0;

int userSongRatings = 0;
int userItemRatings = 0;
int djSongRatings = 0;
int djItemRatings = 0;

for (Enumeration e = ratings.elements(); e.hasMoreElements() ;)
{
    cached = (CachedRating) e.nextElement();

    djID    = cached.userID;
    itemID  = cached.itemID;
    rating  = cached.rating;
    type    = cached.type;

    // 0 is not a valid userID
    // ratings < 0 mean it was unrated
    if (djID != 0 || rating < 0)
    {

        if (type == Constants.ITEM_TYPE_SONG)
        {
            songRatings++;

            // store the user's rating
            if (userID == djID)
            {

                userSongRatings++;

                if (rating == 0)
                {
                    songs.initSong(itemID, Song.EXCLUDED);
                    info = (SongInfo) cache.get(itemID,
SongInfoCache.TYPE_SONG);

                    addToAverage(info, 0);
                }
                else
                {

                    data = songs.initSongGetData(itemID,
songType);

                    if (data != null)
                    {

                        info = (SongInfo) cache.get(itemID,
SongInfoCache.TYPE_SONG);

                        // if the song isn't in the cache,
                        it's not encoded

                        // and we can't play it
                        if (info == null)

```

```

Song.EXCLUDED);
{
    songs.initSong(itemID,
}
else
{
    data.setInfo(info);
    data.querySource =
    data.rating.set(rating,
    // add this rating to all
    addToAverage(info, rating);
}
}
}
else // this is another user's song rating
{
    djSongRatings++;
    data = songs.initSongGetData(itemID,
Song.UNRATED);
    if (data != null)
    {
        data.querySource = SongData.SOURCE_DJS;
        data.djsAverage.add(rating);
    }
}
}
// don't count various artists ratings
else if (!(type == Constants.ITEM_TYPE_ARTIST &&
ArtistInfo.isVariousArtists(itemID)))
{
    itemRatings++;
    theItem = profile.put(itemID);
    if (djID == userID)
    {
        userItemRatings++;
        theItem.userRating.set(rating);
    }
    else
    {
        djItemRatings++;
        theItem.djsAverage.add(rating);
    }
}

```

```

    }
    }

    rowCount++;
}

Util.out(out, Thread.currentThread().getName() + " added "
        + songRatings + " song ratings ("
        + userSongRatings + " user, "
        + djSongRatings + " dj) "
        + "and " + itemRatings + " item ratings ("
        + userItemRatings + " user, "
        + djItemRatings + " dj)"
        );
Util.printElapsedTime(Thread.currentThread().getName(), startDate);
}

private void addToAverage(SongInfo info, int rating)
{
    if (info != null)
    {
        (profile.put(info.album.artist.ID)).songAverage.add(rating);
    }
}

private String userCriteria()
{
    if (djs.size() <= 0)
        return " = " + userID;

    return "IN (" + userID + ", " + djs.inList() + ")";
}

}

```

D:\My Documents\email\Launch\PlaylistGenerator\GetRatings.java Page 4 of 4 11/05/99 1:35 PM

```

package com.launch.PlaylistGenerator;

import java.util.*;
import java.net.*;

/**
 **/

public final class GetRatingsCacheUsers extends Thread
{
    private static int SLEEP_TIME = (10 * 60 * 1000); // update every 10
minutes

    private static int EXPECTED_TOP_USER_SIZE = 100;

    private GetRatingsCacheUsersInterface personToNotify;

    private static final int UPDATE_DB_CACHED_USERS_SLEEP_COUNT = 6 * 8; //
three times every day (6*8*SLEEP_TIME)

    //-----

    /**
     * @param personToNotify must not be null.
     **/
    public GetRatingsCacheUsers(GetRatingsCacheUsersInterface
personToNotify)
    {
        this.personToNotify = personToNotify;
    }

    public void run()
    {
        Thread.currentThread().setName("GetRatingsCacheUsers");
        Util.debug(Thread.currentThread().getName() + " thread started");
        DBConnection conn;
        String myIP;
        DBResultSet rs;
        Vector v;
        Date benchmark_date;

        try
        {
            myIP = InetAddress.getLocalHost().getHostAddress();
            int update_db_users_list =
UPDATE_DB_CACHED_USERS_SLEEP_COUNT;

            while (personToNotify != null)
            {
                benchmark_date = new Date();
                v = new Vector(EXPECTED_TOP_USER_SIZE);
                conn = new DBConnection();
                rs = conn.executeQuery("exec sp_lcGetUsersToCache_isxd
'" + myIP + "'");

```

```

while (!rs.getBOF() && !rs.getEOF())
{
    v.addElement(new Integer(rs.getInt("userID")));
    rs.next();
}

personToNotify.updateCachedUsers(v);

Util.printElapsedTime(Thread.currentThread().getName() + ", get " + v.size()
+ " rows", benchmark_date);

Thread.sleep(SLEEP_TIME);

//---

if (update_db_users_list <= 0)
{
    // do the update

    Util.debug(new Date().toString() + " Updating
RatingsCacheUserList");

    try
    {
        Hashtable h =
personToNotify.getMostFrequentlyUsedUsers(EXPECTED_TOP_USER_SIZE);

        if (h != null && h.size() > 0)
        {
            String the_command = "exec
sp_lcDeleteRatingsCacheUsers_xxxx";

            conn.executeSQL(the_command);

            Enumeration e = h.keys();

            while (e.hasMoreElements())
            {
                the_command = "exec
sp_lcAddRatingsCacheUser_ixxx " + e.nextElement();

                conn.executeSQL(the_command);
            }

            conn.close();
        }
        catch (DBException dbe)
        {
            System.err.println(new Date().toString()
+ " DBException in GetRatingsCacheUsers: " + dbe.toString());
            dbe.printStackTrace();
        }

        update_db_users_list =

```

```

UPDATE_DB_CACHED_USERS_SLEEP_COUNT;
    }
    else
    {
        Util.debug("update_db_users_list is " +
update_db_users_list);
        update_db_users_list--;
    }

    //---
    conn.close();
}
}
catch (Exception e)
{
    System.err.println(new Date().toString() + " Fatal
Exception in GetRatingsCacheUsers: " + e.getMessage());
    e.printStackTrace();
}

    Util.debug(Thread.currentThread().getName() + " thread done");
}

}

```

D:\My Documents\email\Launch\PlaylistGenerator\GetRatingsCacheUsers.java Page 3 of 3 11/05/99 1:23 PM


```

package com.launch.PlaylistGenerator;

import java.util.*;

public interface GetRatingsCacheUsersInterface
{
    /**
     * @param topUsers will be a vector of Integers, where each integer is a
    userID
     */
    public void updateCachedUsers(Vector topUsers);

    /**
     * This method will return a hash of (Integer USERID, Integer Requests)
     * @param i is the number of users to get
     * @return null if no statistics
     */
    public Hashtable getMostFrequentlyUsedUsers(int i);
}

```

D:\My Documents\email\Launch\PlaylistGenerator\GetRatingsCacheUsersInterface.java Page 1 of 1
11/05/99 1:28 PM

```

package com.launch.PlaylistGenerator;

import java.util.Date;

public class GetRecentlyPlayed extends Thread
{
    Population songs;
    int userID;

    public GetRecentlyPlayed(Population songs, int userID)
    {
        this.songs = songs;
        this.userID = userID;
    }

    public void run()
    {
        Date startDate = new Date();
        Thread.currentThread().setName("GetRecentlyPlayed");

        int rowCount = 0;

        String sql = new String("exec sp_lcGetRecentlyPlayedSongs_xxxx "
                                + userID);

        int songID, albumID, artistID;

        try
        {
            DBConnection conn = new DBConnection();

            DBResultSet rs = conn.executeQuery(sql);

            while(!rs.getBOF() && !rs.getEOF())
            {
                // returns songID, albumID, artistID, lastPlayed

                albumID = rs.getInt("albumID");
                songID = rs.getInt("songID");
                artistID = rs.getInt("artistID");

                // don't play these songs so soon again
                songs.initSong(songID, Song.EXCLUDED);

                songs.artistCounts.increment(artistID);
                songs.albumCounts.increment(albumID);

                rs.next();
                rowCount++;
            }

            conn.close();

```

```

    }
    .catch (DBException oops)
    {
        Util.debug("DBException: " + oops.getMessage());
    }

    Util.debug(Thread.currentThread().getName() + " added " + rowCount + "
songs");
    Util.printElapsedTime(Thread.currentThread().getName(), startDate);
}

```

```

}
D:\My Documents\email\Launch\PlaylistGenerator\GetRecentlyPlayed.java Page 2 of 2 11/05/99 1:26 PM

```

```

package com.launch.PlaylistGenerator;

import java.util.*;
import java.io.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;

/**
 *-----
 *
 * GetSongInfoServlet
 * @author Jeff Boulter
 *-----
 */
public class GetSongInfoServlet extends HttpServlet
{
    public static final byte ONLINE_TIMEOUT = 10;

    //-----

    /**
     * Handle requests...
     */
    public void doGet (
        HttpServletRequest request,
        HttpServletResponse response
    ) throws ServletException, IOException
    {
        String userID;
        String volume;
        String djID;
        String djName;
        String djPossessive;

        String songName = "";
        String albumName = "";
        String artistName = "";
        int songID = 0;
        int albumID = 0;
        int artistID = 0;
        int commRating = 0;
        Date dateAdded = new Date();
        byte origin = 0;
        int mediaID = 0;
        int year = 0;
        int songRating = -1;
        int albumRating = -1;
        int artistRating = -1;

        // get stream for output

```

```

ServletOutputStream out;
response.setContentType("text/html");
out = response.getOutputStream();
response.setHeader("Pragma", "no-cache");
response.setHeader("Cache-control", "no-cache");
response.setHeader("Expires", "0");

try
{

    userID = request.getParameter("rater");

    if (userID == null)
    {
        out.println("no userID passed");
        return;
    }

    DBConnection conn = new DBConnection();

    djID = request.getParameter("djID");
    djName = request.getParameter("djName");

    if (djID == null || djID.equals(userID))
    {
        djName = "You";
        djPossessive = "Your";
    }
    else
    {
        djPossessive = djName + "'s";
    }

    DBResultSet rs = conn.executeQuery("exec
sp_lcGetPlayingInfoForUser_xsxx " + userID);

    while (!rs.getBOF() && !rs.getEOF())
    {
        songName = rs.getString("song");
        albumName = rs.getString("album");
        artistName = rs.getString("artist");
        songID = rs.getInt("songID");
        albumID = rs.getInt("albumID");
        artistID = rs.getInt("artistID");
        commRating = rs.getInt("commRating");
        if (commRating <= 0) { commRating = -1; }
        origin = (byte) rs.getInt("origin");
        mediaID = rs.getInt("mediaID");
        year = rs.getInt("year");

        dateAdded = rs.getTimestamp("dateAdded");

        songRating = rs.getInt("songRating");
    }
}

```

```

        albumRating = rs.getInt("albumRating");
        artistRating = rs.getInt("artistRating");

        rs.next();
    }

    int exclusive = isExclusive(albumName);
    int newStatus = isNew(dateAdded);
    int popular = isPopular(commRating);

    String djs = "";

    if (origin == SongData.SOURCE_DJS_ALBUM)
        djs = djRatings(conn, userID, albumID,
Constants.ITEM_TYPE_ALBUM);
    else if (origin == SongData.SOURCE_DJS_ARTIST)
        djs = djRatings(conn, userID, artistID,
Constants.ITEM_TYPE_ARTIST);
    else
        djs = djRatings(conn, userID, songID,
Constants.ITEM_TYPE_SONG);

    out.print(
        "media_id=" + mediaID + "&"
        + "song_id=" + songID + "&"
        + "song_name=" + escape(songName) + "&"
        + "album_id=" + albumID + "&"
        + "album_name=" + escape(albumName +
formatAlbumYear(year)) + "&"
        + "artist_id=" + artistID + "&"
        + "artist_name=" + escape(artistName) + "&"

        + "exclusive=" + exclusive + "&"
        + "comm_rating=" + commRating + "&"
        + "new=" + newStatus + "&"
        + "origin=" +
escape(SongData.originText(origin, djName, djPossessive)) + "&"
        + "popular=" + popular + "&"

        + "song_rating=" + songRating + "&"
        + "song_rating_type=1" + "&"
        + "album_rating=" + albumRating + "&"
        + "album_rating_type=1" + "&"
        + "artist_rating=" + artistRating + "&"
        + "artist_rating_type=1"

        + djs

        + fans(conn, songID)
        + radioStations(conn, userID, songID)

        + "&ticker_text=&image_url=" // not used
    );

```

```

        volume = request.getParameter("volume");
        saveVolume(conn, userID, volume);

        conn.close();
    }
    catch (DBException e)
    {
        System.err.println("DBException: " + e.getMessage());
        e.printStackTrace();
    }
    catch (Exception e)
    {
        out.println("Exception raised: " + e);
        e.printStackTrace();
    }

    out.close();
}

private void saveVolume(DBConnection conn, String userID, String
volumeStr) throws DBException
{
    if (volumeStr == null)
        return;

    double volume = 0;

    try
    {
        Double dblVolume = new Double(volumeStr);

        if (dblVolume != null)
            volume = dblVolume.doubleValue();
    }
    catch (Exception e)
    {
        return;
    }

    if (volume > 0 && volume <= 100)
    {
        conn.executeSQL("exec sp_lcSetVolume_isux " + userID + ", "
+ volume);
    }
}

private String djRatings(DBConnection conn, String userID, int itemID,
String storedProc, String variableName) throws DBException
{

```

```

        String result = "";

        String djName;
        String ratingStr;
        int rating;
        int count = 1;

        DBResultSet rs = conn.executeQuery("exec " + storedProc + " " +
        userID + ", " + itemID);

        while (!rs.getBOF() && !rs.getEOF())
        {
            rating = rs.getInt("rating");

            if (rating <= 0)
            {
                ratingStr = "X";
            }
            else
            {
                ratingStr = "" + rating;
            }

            result = result.concat(
                "&" + variableName + "_name" + count + "=" +
            escape(rs.getString("alias"))
                + "&" + variableName + "_id" + count + "=" +
            rs.getInt("userID")
                + "&" + variableName + "_value" + count + "=" +
            ratingStr
                + "&" + variableName + "_online" + count + "=" +
            isOnline(rs.getInt("minutesSincePlay"))
            );

            count++;
            rs.next();
        }

        return result;
    }

    private String djRatings(DBConnection conn, String userID, int itemID,
    byte itemType) throws DBException
    {
        if (itemType == Constants.ITEM_TYPE_SONG)
        {
            return djRatings(conn, userID, itemID,
            "sp_lcGetUserDJRatingsForSongID_xxxx", "dj_rating");
        }
        else if (itemType == Constants.ITEM_TYPE_ALBUM)
        {
            return djRatings(conn, userID, itemID,
            "sp_lcGetUserDJRatingsForAlbumID_xxxx", "dj_rating");
        }
        else if (itemType == Constants.ITEM_TYPE_ARTIST)
    }

```



```

        {
            return djRatings(conn, userID, itemID,
"sp_lcGetUserDJRatingsForArtistID_xsxx", "dj_rating");
        }

        return "";
    }

```

```

private String radioStations(DBConnection conn, String userID, int
songID) throws DBException

```

```

{
    int count = 0;
    String result = "";

    DBResultSet rs = conn.executeQuery("exec
sp_lcGetSubscribedBDSStationsPlayingSong_xsxx " + userID + ", " + songID);

    while (!rs.getBOF() && !rs.getEOF())
    {
        result = result.concat(
            "&radio_id" + count + "=" +
rs.getInt("bdsStationID")
            + "&radio_name" + count + "=" +
escape(rs.getString("callLetters") + " " + rs.getString("description"))
        );

        count++;
        rs.next();
    }

    return result;
}

```

```

private String fans(DBConnection conn, int songID) throws DBException
{

```

```

    String result = "";

    int count = 1;
    int rating;
    String ratingStr = "";

    DBResultSet rs = conn.executeQuery("exec sp_lcGetFans_xsxx " +
songID);

    while (!rs.getBOF() && !rs.getEOF() && count <= 5)
    {
        result = result.concat(
            "&fan_name" + count + "=" +
escape(rs.getString("alias"))
            + "&fan_id" + count + "=" + rs.getInt("userID")
            + "&fan_online" + count + "=" +
isOnline(rs.getInt("minutesSincePlay"))
        );
    }
}

```

```

        );

        count++;
        rs.next();
    }

    if (count > 1 && !rs.getEOF())
    {
        result = result.concat("&fan_id" + count + "=0" +
"&fan_name" + count + "=more...");
    }

    return result;
}

```

```

private String formatAlbumYear(int year)
{
    if (year > 0)
    {
        return " (" + year + ")";
    }

    return "";
}

```

```

private int isExclusive(String albumName)
{
    if (albumName != null)
    {
        if (albumName.indexOf("Launch Live") > -1)
        {
            return 1;
        }
    }

    return 0;
}

```

```

private int isOnline (int lastPlay)
{
    if (ONLINE_TIMEOUT > lastPlay)
    {
        return 1;
    }

    return 0;
}

```

```

private int isPopular (int commRating)
{

```

```

        if (commRating > Constants.POPULAR_THRESHOLD)
        {
            return 1;
        }

        return 0;
    }

    private int isNew (Date dateAdded)
    {
        if (dateAdded == null)
        {
            return 0;
        }

        long twoWeeks = Util.MILLISECONDS_IN_SECOND *
                           Util.SECONDS_IN_MINUTE *
                           Util.MINUTES_IN_HOUR *
                           Util.HOURS_IN_DAY *
                           14;

        Date now = new Date();

        if (now.getTime() - dateAdded.getTime() < twoWeeks)
        {
            return 1;
        }

        return 0;
    }

    private String escape(String thing)
    {
        if (thing == null)
        {
            return "";
        }

        return URLEncoder.encode(thing);
    }

    public void init (ServletConfig config)
        throws ServletException
    {
        super.init(config);
    }

    public void destroy()
    {
    }
}

```

/* eof */

```

package com.launch.PlaylistGenerator;

import java.util.*;

public final class GetSongRatingsFromDB extends Thread
{
    private Vector userIDs;
    private Vector results;

    //-----

    public GetSongRatingsFromDB(Vector userIDs, Vector results)
    {
        this.userIDs = userIDs;
        this.results = results;
    }

    public void run()
    {
        Thread.currentThread().setName("GetSongRatingsFromDB");
        Util.debug(Thread.currentThread().getName() + " thread started");
        Date startDate = new Date();

        try
        {
            String sql = "SELECT iUserID_FK userID, iSongID_FK songID,
iRating rating FROM a200SongRating WHERE iUserID_FK IN (" +
RatingsCache.GetVectorAsCommaDelimitedList(userIDs) + ')';

            DBConnection conn = new DBConnection();
            DBResultSet rs = conn.executeSQL(sql);
            CachedRating cr;

            while (!rs.getBOF() && !rs.getEOF())
            {
                cr = new CachedRating(rs.getInt("userID"),
rs.getInt("songID"), (byte)rs.getInt("rating"), Constants.ITEM_TYPE_SONG);
                results.addElement(cr);
                rs.next();
            }

            conn.close();
        }
        catch (DBException oops)
        {
            System.err.println("DBException in GetSongRatingsFromDB: "
+ oops.getMessage());
        }

        Util.printElapsedTime(Thread.currentThread().getName(),
startDate);
    }
}

```

```

package com.launch.PlaylistGenerator;

import java.util.Hashtable;

/**
 * A hashtable that uses ints as keys and values.
 */
public class IntHash extends Hashtable
{
    public synchronized int get(int key)
    {
        Object thing = get(new Integer(key));

        if (thing == null)
            return 0;
        else
            return ((Integer) thing).intValue();
    }

    public synchronized int put(int key, int value)
    {
        put(new Integer(key), new Integer(value));

        return value;
    }

    private synchronized int change(int key, int valueChange)
    {
        return put(key, get(key) + valueChange);
    }

    public synchronized int increment(int key)
    {
        return change(key, 1);
    }

    public synchronized int decrement(int key)
    {
        return change(key, -1);
    }

    public synchronized int increment(int key, int howMuch)
    {
        return change(key, howMuch);
    }

    public synchronized int decrement(int key, int howMuch)
    {
        return change(key, -howMuch);
    }
}

```

```

package com.launch.PlaylistGenerator;

import java.util.Hashtable;
import java.util.Enumeration;
import javax.servlet.ServletOutputStream;

public class ItemsProfile
{
    private Hashtable hash;

    public ItemsProfile()
    {
        hash = new Hashtable();
    }

    public synchronized Item get(int itemID)
    {
        return get(new Integer(itemID));
    }

    public synchronized Item get(Integer itemID)
    {
        return (Item) hash.get(itemID);
    }

    /**
     * puts a new item in the hash and returns it.
     * If it's already there, just return it
     */
    public synchronized Item put(int itemID)
    {
        Integer ID = new Integer(itemID);

        Item it = get(ID);

        if (it == null)
        {
            it = new Item(itemID);
            hash.put(ID, it);
            return it;
        }
        else
            return it;
    }

    public void print(ServletOutputStream out, SongInfoCache cache)
    {
        for (Enumeration e = hash.keys(); e.hasMoreElements(); ) {
            Item anItem = get((Integer) e.nextElement());

            Util.out(out, anItem.toString(cache));
        }
    }
}

```

```

    }

    public String inList(byte type)
    {
        String list = "";

        for (Enumeration e = hash.keys(); e.hasMoreElements() ;) {
            Item anItem = get((Integer) e.nextElement());
            if (type == Item.TYPE_ANY || anItem.getType() == type)
            {
                list = list.concat(anItem.itemID + ",");
            }
        }

        // remove that extra comma
        if (list.length() > 0)
            list = list.substring(0, list.length() - 1);

        return list;
    }
}

```

D:\My Documents\email\Launch\PlaylistGenerator\ItemsProfile.java Page 2 of 2 11/05/99 1:32 PM

```

package com.launch.PlaylistGenerator;

public class Item
{
    public final static byte TYPE_ANY      = 0;
    public final static byte TYPE_ALBUM    = 1;
    public final static byte TYPE_ARTIST   = 2;
    public final static byte TYPE_UNKNOWN  = 10;

    public int itemID;
    public Rating userRating;
    private boolean songAvgScoreCalculated = false;

    private double songAvgScore;

    // the average rating from all djs for this item
    public AverageRating djsAverage;

    // average rating of all songs by an artist
    public AverageRating songAverage;

    public double songAverageScore(ArtistInfo info)
    {
        if (!songAvgScoreCalculated)
        {
            songAvgScoreCalculated = true;

            double songsByArtist = Math.min(info.songs.size(),
Constants.MAX_SONGS_BY_ARTIST);
            double songsRated     = Math.min(songAverage.count(),
Constants.MAX_SONGS_BY_ARTIST);

            // deviation from the average
            songAvgScore = ((songAverage.get() - Constants.DEFAULT_RATING)
                * (songsRated / songsByArtist)) + Constants.DEFAULT_RATING;
        }

        return songAvgScore;
    }

    public boolean inGenres = false;

    public byte getType()
    {
        if (itemID == 0)
            return TYPE_UNKNOWN;
        else if (itemID < 1000000)
            return TYPE_ALBUM;
        else
            return TYPE_ARTIST;
    }

    public String typeName()
    {

```



```

        byte type = getType();

        if (type == TYPE_ALBUM)
            return "Album";
        else if (type == TYPE_ARTIST)
            return "Artist";
        else
            return "Unknown";
    }

    public Item()
    {
        userRating = new Rating();
        djsAverage = new AverageRating();
        songAverage = new AverageRating();
    }

    public Item(int itemID)
    {
        this();
        this.itemID = itemID;
    }

    public String toString(SongInfoCache cache)
    {
        String title = "(Not available)";
        byte type = getType();

        if (type == TYPE_ARTIST)
        {
            ArtistInfo artist = (ArtistInfo) cache.get(itemID,
SongInfoCache.TYPE_ARTIST);

            if (artist != null)
                title = artist.title;
        }
        else if (type == TYPE_ALBUM)
        {
            AlbumInfo album = (AlbumInfo) cache.get(itemID,
SongInfoCache.TYPE_ALBUM);

            if (album != null)
                title = album.title;
        }

        return typeName() + " \"" + title + "\" (" + itemID + ") "
            + "user=" + userRating.toString()
            + " djs=" + djsAverage.toString()
            + " songAverage=" + songAverage.toString()
            + " songAvgScore=" + songAvgScore;
    }
}

```

```

package com.launch.PlaylistGenerator;

public class MediaFormat
{
    public final static byte WINDOWSMEDIA = 1;
    public final static byte REALMEDIA   = 2;
    public final static byte QUICKTIME   = 3;

    private boolean beenset = false;

    private byte value;

    // when we start supporting more than one format, just take this out
    public MediaFormat()
    {
        value = WINDOWSMEDIA;
        beenset = true;
    }

    public MediaFormat(byte format)
    {
        value = format;
        beenset = true;
    }

    public byte get()
    {
        return value;
    }

    public void set(byte format)
    {
        value = format;
        beenset = true;
    }

    public boolean isSet()
    {
        return beenset;
    }

    public String toString()
    {
        if (value == WINDOWSMEDIA)
            return "WindowsMedia";
        else if (value == REALMEDIA)
            return "RealMedia";
        else if (value == QUICKTIME)
            return "QuickTime";

        return "UNKNOWN";
    }
}

```

```
package com.launch.PlaylistGenerator;
```

```
import java.io.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
```

```
/**
```

```
*-----
```

```
* PlaylistGeneratorServlet.java 8/16/99
```

```
* Servlet that redirects to media
```

```
* Copyright (c) 1999 Launch, Inc.
```

```
* @author Jeff Boulter
```

```
*-----
```

```
*/
```

→

```
public final class MediaGatewayServlet extends HttpServlet
{
```

```
    /** what browser signature we look for */
```

```
    private static final String mpSignature = "NSPlayer";
```

```
    /** when we get an unauthorized browser, play this */
```

```
    private static final String unauthorizedBrowser =
```

```
"audio/errors/unauthorizedbrowser.asf";
```

```
    /** when we get an unauthorized user, play this */
```

```
    private static final String unauthorizedUser =
```

```
"audio/errors/unauthorizeduser.asf";
```

```
    /** when we get an unauthorized user, play this */
```

```
    private static final String outOfMedia = "audio/errors/outofmedia.asf";
```

```
    /** how many tries we take to get media */
```

```
    private static final int MAX_ITERATIONS = 5;
```

```
    /** this is the header that media player uses to indicate which query it is
```

```
*/
```

```
    private static final String CONTEXT_TAG = "request-context=";
```

```
    /** To work around a problem with reading multiple headers with the same name
in servlet 2.0 + jrun, we look for these headers to determine the context */
```

```
    private static final String FIRST_REQUEST_PRAGMA = "xClientGUID";
```

```
    private static final String SECOND_REQUEST_PRAGMA = "stream-switch-entry";
```

```
    private static final String REQUEST_CONTEXT = "request-context=";
```

```
    private static final int STREAMING_MEDIA_TIMEOUT=1000*60*15;
```

```
    /**
```

```
    * Handle requests...
```

```
    */
```

```
    public final void doGet (HttpServletRequest request, HttpServletResponse
```

response) throws ServletException, IOException

```
{  
  
//      Util.debug("MediaRedirectServlet:doGet() received a request");  
  
      DBConnection conn = null;  
      ServletOutputStream out = null;  
  
      int context;  
      int userID = -1;  
      boolean debug=false;  
  
      try  
      {  
          // get connections and streams  
          conn = new DBConnection();  
          out = response.getOutputStream();  
  
          // get parameters from http  
          debug = (request.getParameter("ralph") != null);  
  
          // setup response data  
          setResponseHeaders(response);  
          setResponseContentType(response, debug);  
  
          // get parameters from http  
          userID = Integer.parseInt(request.getParameter("u"));  
  
          if (!checkUserAgent(request.getHeader("USER_AGENT"), debug, out))  
          {  
              return;  
          }  
  
          // muck with clip and clip schedule  
          ClipSchedule schedule = new ClipSchedule(userID);  
          schedule.init(conn); //db call 1  
  
          Clip aClip = null;  
          int iteration;  
  
          boolean done = false;  
  
          // keep going until we get a good path  
          for (iteration = 0; iteration < MAX_ITERATIONS && !done;  
iteration++)  
          {  
              aClip = new Clip(schedule.nextClipType(debug, out));  
  
              if (aClip == null || aClip.type() == Clip.TYPE_NONE)  
              {  
                  done = true;  
                  System.err.println("user " + userID + " is out of  
songs to play");  
              }  
          }  
      }  
      catch (Exception e)  
      {  
          Util.debug(e.toString());  
      }  
  }
```

```

    }
    else
    {
        // get the paths and stuff
        aClip.getPath(conn, schedule); // db call 2

        if (aClip.isSet())
        {
            done = true;
        }
        else
        {
            done = true;
            System.err.println("user " + userID + " is out
of media of type " + aClip.typeName() + " to play");
        }
    }

    // update the playlist
    schedule.playlist.save(conn, userID); // db call 3

    if (aClip == null)
        out.println(Constants.STREAM_SERVER + "/" + outOfMedia);
    else
    {
        // log the play
        aClip.logPlay(conn, userID); // db call 4

        // get the URL
        out.println(aClip.URL());
    }
}
catch (NumberFormatException e)
{
    out.println("Bad userId");

    // print out the MMS path to redirect to

    if (debug)
    {
        out.println("redirecting to " + unauthorizedUser);
    }
    else
    {
        out.println(Constants.STREAM_SERVER + "/" +
unauthorizedUser);
    }
}
catch (Throwable e)
{
    System.err.println("Generic Exception in MediaGateway for userID
" + userID + ": " + e.getMessage());
    e.printStackTrace();
}

```

```

    }
    finally
    {
        try
        {
            if (out!=null)
            {
                out.close();
            }

            if (conn!=null)
            {
                conn.close();
            }
        }
        catch (SocketException se)
        {
            // don't do anything, the person disconnected, no error,
            (or mediaplayer sampled first 32 bytes.)
        }
        catch (Exception e1)
        {
            e1.printStackTrace();
        }
    }
}

```

```

    private final boolean checkUserAgent(String agent, boolean debug,
ServletOutputStream out) throws IOException
    {
        if (!(agent!=null && agent.startsWith(mpSignature)))
        {
            if (debug)
            {
                out.println("invalid useragent. Would stream " +
unauthorizedBrowser);
                return true;
            }
            else
            {
                out.println(Constants.STREAM_SERVER + "/" +
unauthorizedBrowser);
            }

            return(false);
        }
        else
        {
            return(true);
        }
    }
}

```

```

    private final void setResponseContentType(HttpServletResponse response,

```

```

boolean debug)
{
    if (debug)
    {
        response.setContentType("text/plain");
    }
    else
    {
        response.setContentType("video/x-ms-asf");
    }
}

private final void setResponseHeaders(HttpServletResponse response)
{
    response.setHeader("Pragma", "no-cache");
    response.setHeader("Cache-control", "no-cache");
    response.setHeader("Expires", "0");
}

/*

private static final void readFileToOutputStream(String filename,
HttpServletResponse response, boolean debug)
{
    readFileToOutputStream(new File(filename), response, debug);
}

private static final void readFileToOutputStream(File the_file,
HttpServletResponse response, boolean debug)
{
    try
    {
        BufferedInputStream bis=new BufferedInputStream(new
FileInputStream(the_file));

        BufferedOutputStream bos=new
BufferedOutputStream(response.getOutputStream());
        bos.flush(); //this is to ward off any problems I think there
might be a jrun problem with initializing the output stream fast enough, i.e.
before we get there...
        BufferedWriter br=new BufferedWriter(new
OutputStreamWriter(bos));

        if (debug)
            Util.out(response.getOutputStream(), "streaming file " +
the_file + " of size " + the_file.length());
        else
            response.setContentLength((int)the_file.length());

        // System.err.println("streaming file " + the_file + " of size "
+ the_file.length());

        RedirectStream redirecting_stream=new RedirectStream(bis, bos,

```

```

debug, response.getOutputStream());

        redirecting_stream.start();

        redirecting_stream.join(STREAMING_MEDIA_TIMEOUT, 0);

        if (redirecting_stream.isAlive()) redirecting_stream.stop();
        //System.err.println("finished streaming");
    }
    catch (SocketException se)
    {
        // don't do anything, the person disconnected, no error, (or
mediaplayer sampled first 32 bytes.)
    }
    catch (FileNotFoundException fe)
    {
        System.err.println("readFileToOutputStream could not find file "
+ the_file + " for reading:" + fe.getMessage());
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}

private int getContext(HttpServletRequest request)
{
    try
    {

        String pragma = request.getHeader("pragma");

//        Util.debug("pragma is " + pragma);

        if (pragma == null)
            return 0;

        int index = pragma.indexOf(REQUEST_CONTEXT);

//        Util.debug("index is " + index);

        if (index < 0)
        {
            return 0;
        }
        else
        {
            int start = index + REQUEST_CONTEXT.length();
            String contextNum = pragma.substring(start, start + 1);
//            Util.debug("contextNum is " + contextNum);
            return Integer.parseInt(contextNum);
        }
    }
}

// when I can read multiple headers with the same name I should use the below code

```



```

//          int location=pragma.indexOf(CONTEXT_TAG);
//          location=location+CONTEXT_TAG.length();

//          int last_location;

//          for (last_location=location; last_location<pragma.length() &&
pragma.charAt(last_location)!=';'; last_location++);
//          return(Integer.parseInt(pragma.substring(location,
last_location)));
    }
    catch (Exception e)
    {
        Util.debug("Exception caught in getContext: " + e.toString());
        return 0;
    }
}

*/
}

```

D:\My Documents\email\Launch\PlaylistGenerator\MediaGatewayServlet.java Page 7 of 7 11/05/99 1:24 PM

```

package com.launch.PlaylistGenerator;

import java.util.Vector;

public class MediaList
{
    private Vector media = new Vector(0, 1);

    public void add(short mediaType, int mediaID, String filepath)
    {
        media.addElement(new Media(mediaID, mediaType, filepath));
    }

    public boolean inType(short mediaType)
    {
        Media test;

        for (int i = 0; i < media.size(); i++)
        {
            test = (Media) media.elementAt(i);

            if (test.mediaType == mediaType)
                return true;
        }

        return false;
    }

    public int getID(short mediaType)
    {
        for (int i = 0; i < media.size(); i++)
        {
            Media aMedia = (Media) media.elementAt(i);

            if (aMedia.mediaType == mediaType)
                return aMedia.mediaID;
        }

        return 0;
    }

    public String getFilePath(short mediaType)
    {
        for (int i = 0; i < media.size(); i++)
        {
            Media aMedia = (Media) media.elementAt(i);

            if (aMedia.mediaType == mediaType)
                return aMedia.filepath;
        }

        return null;
    }
}

```

```

public int size()
{
    return media.size();
}

public Media typeAt(int index)
{
    return (Media) media.elementAt(index);
}

public String toString()
{
    String result = "";

    if (media == null)
        return "(none)";

    for (int i = 0; i < media.size(); i++)
    {
        result = result.concat(media.elementAt(i).toString() + ",");
    }

    return "(" + result + ")";
}

```

```

}
D:\My Documents\email\Launch\PlaylistGenerator\MediaList.java      Page 2 of 2      11/05/99 1:28 PM

```

```

package com.launch.PlaylistGenerator;

public class Media
{
    int mediaID;
    short mediaType;
    String filepath;

    public Media(int mediaID, short mediaType, String filepath)
    {
        this.mediaID = mediaID;
        this.mediaType = mediaType;
        this.filepath = filepath;
    }

    public String toString()
    {
        return mediaType + ": " + mediaID;
    }

    public static short getMediaType(Bandwidth speed, MediaFormat format)
    {
        if (format.get() == MediaFormat.WINDOWSMEDIA)
        {
            if (speed.get() == Bandwidth.SPEED_28)
                return 211;
            else if (speed.get() == Bandwidth.SPEED_56)
                return 147;
            else if (speed.get() >= Bandwidth.SPEED_100)
                return 212;
            else
                return 0;
        }

        return 0;
    }

    public static Bandwidth typeToBandwidth(short mediaType)
    {
        if (mediaType == 211)
            return new Bandwidth(Bandwidth.SPEED_28);
        else if (mediaType == 147)
            return new Bandwidth(Bandwidth.SPEED_56);
        else if (mediaType == 212)
            return new Bandwidth(Bandwidth.SPEED_100);

        return new Bandwidth();
    }
}

```

D:\My Documents\email\Launch\PlaylistGenerator\Media.java Page 1 of 1 11/05/99 1:28 PM

```

package com.launch.PlaylistGenerator;

import javax.servlet.ServletOutputStream;

/**
 */
public class PickCount
{
    int explicit;
    int implicit;
    int unrated;
    String method = "";

    public PickCount(int userID, int djID, int ratio, int playlistSize,
Population songs, ServletOutputStream out)
    {

        float explicitSize = songs.explicit.size();
        float implicitSize = songs.implicit.size();
        float unratedSize = songs.unrated.size();

        Util.out(out, "Available: explicit songs: " + explicitSize + ",
implicit songs: " + implicitSize + ", unrated songs: " + unratedSize);
        Util.out(out, "Ratio: " + ratio);

        // if you're listening to someone else's station, try to not listen to
any unrated songs

        if (userID == djID)
        {
            // let's try to use their ratio

            double totalRated = (explicitSize + implicitSize);

            if (totalRated < Constants.MIN_RATINGS_TO_HONOR_RATIO)
            {
                method = "New User Unrated Ratio";
                ratio = Constants.NEW_USER_UNRATED_RATIO;
            }

            int maxPlicit = (int) Math.round(playlistSize * (100 - ratio) *
0.01);
            int maxRatedToPick = (int) Math.round(explicitSize *
Constants.MAX_PERCENT_RATED_SONGS_TO_PICK * 0.01);

            // pick three times as much from rated
            int explicitToPick = (int) Math.round(playlistSize * (100 -
ratio) * 0.01 * (explicitSize / totalRated) * 3);

            int implicitToPick = maxPlicit - explicitToPick;

            explicit = (int) Math.min(maxRatedToPick, explicitToPick);
            implicit = (int) Math.min(implicitSize, implicitToPick);

            // pick up the slack in unrated

```

```

        unrated = (playlistSize - explicit - implicit);

        method = "Unrated Ratio";
    }
    // if you're listening to someone else's station and they have enough
ratings,
    // don't play unrated
    else if ((explicitSize + implicitSize) >
Constants.MIN_SIZE_FOR_NO_UNRATED)
    {
        explicit = (int) Math.round(playlistSize * 0.50);
        explicit = (int) Math.round(Math.min(explicit, (explicitSize *
Constants.MAX_PERCENT_RATED_SONGS_TO_PICK) * 0.01));

        implicit = (int) Math.min(playlistSize, implicitSize) - explicit;

        method = "DJ play - no unrated";

        // if we didn't get enough, use the default method
        if (explicit + implicit < playlistSize)
        {
            explicit = (int) Math.round(playlistSize * 0.33);
            explicit = (int) Math.round(Math.min(explicit,
(explicitSize * Constants.MAX_PERCENT_RATED_SONGS_TO_PICK) / 100.0));

            implicit = (int) Math.round(playlistSize * 0.33);
            implicit = (int) Math.round(Math.min(implicit,
(implicitSize * Constants.MAX_PERCENT_RATED_SONGS_TO_PICK) / 100.0));

            unrated = playlistSize - explicit - implicit;

            method = "DJ play - not enough rated";
        }
    }
    // if neither of these worked
else
    {
        explicit = (int) Math.round(playlistSize * 0.33);
        explicit = (int) Math.round(Math.min(explicit, (explicitSize *
Constants.MAX_PERCENT_RATED_SONGS_TO_PICK) / 100.0));

        implicit = (int) Math.round(playlistSize * 0.33);
        implicit = (int) Math.round(Math.min(implicit, (implicitSize *
Constants.MAX_PERCENT_RATED_SONGS_TO_PICK) / 100.0));

        unrated = playlistSize - explicit - implicit;

        method = "Default 33/33/33 method";
    }

    Util.out(out, "Picking: explicit songs: "
        + explicit
        + ", implicit songs: "

```

```

        + implicit
        + ", unrated songs: "
        + unrated
        + ", method = " + method
    );
}

public String toString()
{
    return "explicit to pick: "
        + explicit
        + ", implicit to pick: "
        + implicit
        + ", unrated to pick: "
        + unrated;
}

public void reset()
{
    explicit = 0;
    implicit = 0;
    unrated = 0;
}
}

```

D:\My Documents\email\Launch\PlaylistGenerator\PickCount.java Page 3 of 3 11/05/99 1:24 PM

```

package com.launch.PlaylistGenerator;

import java.util.Vector;

public class PickList extends Vector
{
    public PickList(PickCount counts)
    {
        // make a list of all the song types that we need to pick
        for (int i = 0; i < counts.explicit; i++)
            addElement(Song.EXPLICIT);

        for (int i = 0; i < counts.implicit; i++)
            addElement(Song.IMPLICIT);

        for (int i = 0; i < counts.unrated; i++)
            addElement(Song.UNRATED);
    }

    public void addElement(short value)
    {
        addElement(new Short(value));
    }

    public void reAdd (short type, Vector songGroup, Population songs)
    {
        // try to pick from the same bucket again
        if (songGroup.size() > 0)
            addElement(type);
        // otherwise, try the other ones
        else if (songs.explicit.size() > 0)
            addElement(Song.EXPLICIT);
        else if (songs.implicit.size() > 0)
            addElement(Song.IMPLICIT);
        else if (songs.unrated.size() > 0)
            addElement(Song.UNRATED);
    }

    public short getRandom()
    {
        if (size() < 0)
            return 0;

        int lucky = (int) Util.random(size() - 1);

        // figure out what group to pick from
        short type = ((Short) elementAt(lucky)).shortValue();
    }
}

```



```
removeElementAt(lucky);
```

```
return type;
```

```
}
```

```
}
```

D:\My Documents\email\Launch\PlaylistGenerator\PickList.java

Page 2 of 2

11/05/99 1:27 PM

```

package com.launch.PlaylistGenerator;

public class PickStatus
{
    public final static int NOT_PICKED = 0;
    public final static int REJECTED = 2;
    public final static int PICKED = 1;

    int status;
    int order = -1;
    short percentile;

    public String toString()
    {
        return toDisplayString(Util.DISPLAY_TEXT);
    }

    public String toDisplayString(int displayType)
    {
        String redStart = "";
        String greenStart = "";
        String fontEnd = "";

        if (displayType == Util.DISPLAY_HTML)
        {
            redStart = "<FONT COLOR=red><B>";
            greenStart = "<FONT COLOR=green><B>";
            fontEnd = "</B></FONT>";
        }

        switch (status) {
            case NOT_PICKED:
                return "N ";
            case PICKED:
                return greenStart + " P " + fontEnd;
            case REJECTED:
                return redStart + " R " + fontEnd;
            default:
                return " ";
        }
    }
}

```

D:\My Documents\email\Launch\PlaylistGenerator\PickStatus.java

Page 1 of 1 11/05/99 1:26 PM

```

package com.launch.PlaylistGenerator;

import java.util.Enumeration;

public class PlayDataHash extends IntHash
{
    public String toString()
    {
        String myString = "";

        for (Enumeration e = keys(); e.hasMoreElements() ;) {
            // debug.write("iteration " + i++);
            int stationID = ((Integer) e.nextElement()).intValue();
            int rank = get(stationID);
            myString = myString.concat(
                "stationID: " +
                stationID +
                "=" +
                rank +
                "\n");
        }

        return myString;
    }
}

```

D:\My Documents\email\Launch\PlaylistGenerator\PlayDataHash.java Page 1 of 1 11/05/99 1:26 PM

```

package com.launch.PlaylistGenerator;

import java.util.Hashtable;
import java.util.Date;
import java.util.Enumeration;
import java.text.SimpleDateFormat;
import java.io.InputStreamReader;
import java.text.ParsePosition;
import java.io.IOException;
import java.util.Calendar;

public class PlayDates
{
    private static final String dateFormat = "yyyy-MM-dd HH:mm:ss";

    private Hashtable hash;

    int userID;

    double secondsInDay = Util.MILLISECONDS_IN_SECOND *
                           Util.SECONDS_IN_MINUTE *
                           Util.MINUTES_IN_HOUR *
                           Util.HOURS_IN_DAY;

    // for date parsing
    private static StringBuffer year    = new StringBuffer("1234");
    private static StringBuffer month   = new StringBuffer("12");
    private static StringBuffer day      = new StringBuffer("12");

    private static StringBuffer hour     = new StringBuffer("12");
    private static StringBuffer minutes  = new StringBuffer("12");

    public Date dbDate = new Date();

    private boolean loaded = false;

    public PlayDates()
    {
        hash = new Hashtable();
    }

    public void put(int songID, Date lastPlayed)
    {
        // the common case is that they will have NOT played this song before,
        // so create the Integer object in anticipation that we will use it for
        // the put as well.

        Integer i = new Integer(songID);

        Date before = get(i);

        // save only the most recent play of a song
    }

```

```

        if (before == null || before.getTime() < lastPlayed.getTime())
        {
            hash.put(i, lastPlayed);
        }
    }

    public Date get(int songID)
    {
        return (Date) hash.get(new Integer(songID));
    }

    public Date get(Integer songID)
    {
        return (Date) hash.get(songID);
    }

    public Enumeration keys()
    {
        return hash.keys();
    }

    public void remove(Integer songID)
    {
        hash.remove(songID);
    }

    public int size()
    {
        return hash.size();
    }

    public String toString()
    {
        String result = "";

        for (Enumeration e = hash.keys(); e.hasMoreElements() ;) {
            Integer songID = (Integer) e.nextElement();
            Date playedAt = get(songID);
            result = result.concat("{ " + songID + " = " + playedAt + " } ");
        }

        return result;
    }

    public String toDBString()
    {
        Date startDate = new Date();

        StringBuffer buffer = new StringBuffer(100000);

        Calendar cal = Calendar.getInstance();

        Integer songID;

```

```

        Date playedAt;

        for (Enumeration e = hash.keys(); e.hasMoreElements() ;) {
            songID = (Integer) e.nextElement();
            playedAt = get(songID);

//            System.out.println(playedAt);

            cal.setTime(playedAt);

            buffer.append(cal.get(Calendar.YEAR) + "-"
                + leadingZero(cal.get(Calendar.MONTH) + 1) +
                + leadingZero(cal.get(Calendar.DAY_OF_MONTH))
                + leadingZero(cal.get(Calendar.HOUR_OF_DAY))
                + leadingZero(cal.get(Calendar.MINUTE)) +
                + leadingZero(cal.get(Calendar.SECOND)) +
                + ":00=" + songID + ",");

//            result = result.concat(formatter.format(playedAt) + "=" + songID
//            + ",");
        }

        Util.printElapsedTime("toDBString", startDate);

        return buffer.toString();
    }

    public static final String leadingZero (int value)
    {
        if (value < 10)
            return "0" + value;

        return value + "";
    }

    public float getScore(Integer songID)
    {
        Date lastPlayed = get(songID);

        if (lastPlayed == null)
            return 0;

        double secondsSincePlayed = new Date().getTime() -
lastPlayed.getTime();
        double daysSincePlayed = secondsSincePlayed / secondsInDay;
        double logValue = Math.log(daysSincePlayed + 0.01);
        return (float) Math.min(100, (22.0 * logValue));
    }

    public void save(DBConnection conn)
    {
//        Date dateStarted = new Date();

```

```

        if (!loaded)
            return;

        try
        {
            conn.executeUpdate("exec sp_lcSavePlayHistoryText_isux " + userID
+ ", '" + toString() + "'", false);
        }
        catch (DBException e)
        {
            System.err.println("DBException in PlayDates:save: " +
e.toString());
        }

//        Util.printElapsedTime("save", dateStarted);
    }

    public void markRecentlyPlayed(SongInfoCache cache, Population songs)
    {

        double now = dbDate.getTime();
        double lastThreeHours = Util.MILLISECONDS_IN_SECOND *
                                Util.SECONDS_IN_MINUTE *
                                Util.MINUTES_IN_HOUR *
                                3;

        Integer songID;
        Date playedAt;
        SongInfo info;
        int artistID, albumID;

        for (Enumeration e = hash.keys(); e.hasMoreElements() ;)
        {
            songID = (Integer) e.nextElement();
            playedAt = get(songID);

            if (now - playedAt.getTime() < lastThreeHours)
            {

                // mark songs played in the last three hours
                // so as to comply with the RIAA rules
                // and make sure we don't pick too many later

                info = (SongInfo) cache.get(songID,
SongInfoCache.TYPE_SONG);

                if (info != null)
                {
                    artistID = info.getArtistID();
                    albumID = info.getAlbumID();

                    // "various artists" albums don't count

                    if (!ArtistInfo.isVariousArtists(artistID))

```

```

        {
            songs.artistCounts.increment(artistID);
        }
        songs.albumCounts.increment(albumID);
    }
}

public void oldLoad(DBConnection conn, int userID)
{
    this.userID = userID;

    try
    {
        String sql = "exec sp_lcoGetLastPlayed_xxxx " + userID;
        DBResultSet rs = conn.executeQuery(sql);

        loaded = true;

        Date lastDate;
        int songID;

        while (!rs.getBOF() && !rs.getEOF())
        {
            songID = rs.getInt("songID");
            lastDate = rs.getTimestamp("lastPlayed");

            put(songID, lastDate);

            rs.next();
        }
    }
    catch (DBException e)
    {
        System.err.println("DBException in PlayDates.oldLoad: " +
e.toString());
    }
}

public void load(DBConnection conn, int userID)
{
    Date startDate = new Date();

    // be careful of the SQL Server TEXTSIZE parameter which is by default
64KB

    this.userID = userID;

```



```

double aDay = Util.MILLISECONDS_IN_SECOND *
               Util.SECONDS_IN_MINUTE *
               Util.MINUTES_IN_HOUR *
               Util.HOURS_IN_DAY;

double aMonth = aDay * Util.DAYS_IN_MONTH;

try
{
    String sql = "exec sp_lcGetSongHistoryText_xsxx " + userID;
    DBResultSet rs = conn.executeSQL(sql);
    Util.printElapsedTime("LP: ran getsonghistorytext", startDate);
    if (!rs.getBOF() && !rs.getEOF())
    {
        loaded = true;
        char[] stuff = new char[100000];

        InputStreamReader reader = new
        InputStreamReader(rs.getAsciiStream("played"));

        Util.printElapsedTime("LP: created reader", startDate);

        dbDate = rs.getTimestamp("dbDate");
        long dbDateTime = dbDate.getTime();

        reader.read(stuff);

        Util.printElapsedTime("LP: read into stuff", startDate);

        Calendar cal = Calendar.getInstance();

        int lastStart = 0;
        int songID = 0;

        // SimpleDateFormat formatter1 = new
        SimpleDateFormat(PlayDates.dateFormat);
        // ParsePosition pos = new ParsePosition(0);

        Date datePlayed = null;
        String parseme = new String();

        long length = stuff.length;

        for (int i = 0; i < length; i++)
        {
            switch (stuff[i])
            {
                case '=':

```

```

        //
lastStart);
        //
        //

//
//
//
//
//
pos);
//

        parseme = new String(stuff, lastStart, i -
pos.setIndex(0);
datePlayed = formatter1.parse(parseme, pos);

datePlayed = parseDate(stuff, lastStart, cal);
System.out.println("date is " + datePlayed);
if (datePlayed == null)
{
    pos.setIndex(0);
    datePlayed = formatter2.parse(parseme,
}
lastStart = i + 1;
break;

case ',':
    parseme = new String(stuff, lastStart, i -

    try
    {
        songID = Integer.parseInt(parseme);
    }
    catch (NumberFormatException e) { }

    // save 'em
    // also don't save them if they're > 30 days

old
    if (songID > 0 && datePlayed != null &&
((dbDateTime - datePlayed.getTime()) < aMonth))
    {
        put(songID, datePlayed);
    }
    songID = 0; // reset
    datePlayed = null; // reset

    lastStart = i + 1;
    break;

case 0:
    // we're at the end of the string
    Util.printElapsedTime("LP: found null at char "

    return;

}

}

}
catch (DBException oops)
{
    Util.debug("DBException in PlayDates.load: " +
oops.getMessage());
}
catch (IOException oops)

```

```

        {
            Util.debug("IOException in PlayDates.load: " +
oops.getMessage());
        }
    }
    /**
     * Why? Because SimpleDateFormat is *way* too slow.
     */
    private final Date parseDate(char[] chars, int start, Calendar cal)
    {
        // 1999-10-13 17:19:00
        // 0123456789012345678

        /*
        String year, month, day, hour, minutes;

        year    = new String(chars, start, 4);
        month    = new String(chars, start + 5, 2);
        day      = new String(chars, start + 8, 2);

        hour     = new String(chars, start + 11, 2);
        minutes  = new String(chars, start + 14, 2);
        */

        year.setCharAt(0, chars[start + 0]);
        year.setCharAt(1, chars[start + 1]);
        year.setCharAt(2, chars[start + 2]);
        year.setCharAt(3, chars[start + 3]);

        month.setCharAt(0, chars[start + 5]);
        month.setCharAt(1, chars[start + 6]);

        day.setCharAt(0, chars[start + 8]);
        day.setCharAt(1, chars[start + 9]);

        hour.setCharAt(0, chars[start + 11]);
        hour.setCharAt(1, chars[start + 12]);

        minutes.setCharAt(0, chars[start + 14]);
        minutes.setCharAt(1, chars[start + 15]);

        int yearInt = 0, monthInt = 0, dayInt = 0, hourInt = 0, minutesInt = 0;

        // try
        // {
            yearInt    = parseInt(year);
            monthInt   = parseInt(month);
            dayInt     = parseInt(day);

            hourInt    = parseInt(hour);
            minutesInt = parseInt(minutes);
        // }
        // catch (NumberFormatException e) { return null;}

        // cal.clear();

```

```

        cal.set(yearInt, monthInt - 1, dayInt, hourInt, minutesInt, 0);
        return cal.getTime();
    }

    private static final int parseInt(StringBuffer s)
    {
        int result = 0;
        int last = s.length() - 1;

        for (int i = last; i >= 0; i--)
        {
            result += char2int(s.charAt(i)) * Math.pow(10, last - i);
        }

        return result;
    }

    private final static int char2int(char ch)
    {
        switch (ch)
        {
            case '1':
                return 1;
            case '2':
                return 2;
            case '3':
                return 3;
            case '4':
                return 4;
            case '5':
                return 5;
            case '6':
                return 6;
            case '7':
                return 7;
            case '8':
                return 8;
            case '9':
                return 9;
            default:
                return 0;
        }
    }
}

```

D:\My Documents\email\Launch\PlaylistGenerator\PlayDates.java Page 9 of 9 11/05/99 1:35 PM

```

package com.launch.PlaylistGenerator;

import java.util.Vector;
import java.util.Hashtable;
import java.util.Enumeration;
import java.util.Date;

public class Playlist
{
    Vector media;
    Vector news;
    Vector ads;
    Vector tips;

    int ID;
    int userID;
    int djID;
    int moodID;
    short mediaType;
    boolean debug;

    boolean popularOnly = false;

    PickCount counts;

    public final static int BUCKET_COUNT = 5;

    private int lastIndex;

    int buckets[];

    IntHash artists;
    IntHash albums;

    public Playlist()
    {
        artists = new IntHash();
        albums = new IntHash();
        counts = null;
        media = new Vector();
        news = new Vector();
        ads = new Vector();
        tips = new Vector();
        buckets = new int[BUCKET_COUNT];

        lastIndex = -1;
        debug = false;
    }

    public Playlist(int playlistID)
    {
        this();
        ID = playlistID;
    }
}

```

```

public void resetSources()
{
    for (int i = 0; i < BUCKET_COUNT; i++)
        buckets[i] = 0;
}

private void saveOrigins(DBConnection conn)
{
    String listString = "";
    SongData data;

    for (int i = 0; i < media.size(); i++)
    {
        listString = listString.concat(((SongData)
media.elementAt(i)).originTclList());
    }

    try
    {
        conn.executeUpdate("exec sp_lcSaveOrigins_ixxd " + userID + ", '" +
listString + "'");
    }
    catch (DBException oops)
    {
        Util.debug("DB Exception: " + oops.getMessage());
    }
}

public Playlist2 toPlaylist2()
{
    Playlist2 result = new Playlist2();

    // copy playlist

    for (int i = 0; i < media.size(); i++)
    {
        result.songs.addElement(((SongData)
media.elementAt(i)).toPlaylistEntry(mediaType));
    }

    // copy news

    for (int i = 0; i < news.size(); i++)
    {
        result.news.addElement(((Clip)
news.elementAt(i)).toPlaylistEntry(mediaType));
    }

    // copy ads

    for (int i = 0; i < ads.size(); i++)
    {
        result.ads.addElement(((Clip)

```

```

ads.elementAt(i)).toPlaylistEntry(mediaType));
    }

    // copy tips
    for (int i = 0; i < tips.size(); i++)
    {
        result.tips.addElement(((Clip)
tips.elementAt(i)).toPlaylistEntry(mediaType));
    }

    return result;
}

public String toString()
{
    IntHash artistCount      = new IntHash();
    IntHash albumCount       = new IntHash();
    IntHash querySource       = new IntHash();
    Hashtable querySourceName = new Hashtable();
    IntHash originSource      = new IntHash();
    Hashtable originSourceName = new Hashtable();

    Hashtable artistNames     = new Hashtable();
    Hashtable albumNames      = new Hashtable();

    String result = "Playlist " + ID + " for userID " + userID
        + " (djID " + djID + ") in mood " + moodID
        + " with mediaType " + mediaType
        + ", pickCounts: " + counts
        + " has " + media.size() + " songs:"
        + Util.newLine;

    for (int i = 0; i < media.size(); i++)
    {
        SongData data = (SongData) media.elementAt(i);
        String songStr = data.getMediaID(mediaType) + " "
            + data.getAlbumID() + " "
            + data.getArtistID() + " "
            + data.songID + " "
            + data.getArtistName() + " "
            + data.getAlbumName() + " "
            + data.getSongName() + Util.newLine;

        querySource.increment(data.querySource);
        querySourceName.put(new Integer(data.querySource),
data.sourceString(data.querySource));

        byte origin = data.origin();
        originSource.increment(origin);
        originSourceName.put(new Integer(origin),
data.sourceString(origin));
    }
}

```

```

        artistCount.increment(data.getArtistID());
        albumCount.increment(data.getAlbumID());

        if (data.getArtistName() != null)
            artistNames.put(new Integer(data.getArtistID()),
data.getArtistName());

        if (data.getAlbumName() != null)
            albumNames.put(new Integer(data.getAlbumID()),
data.getAlbumName());

        result = result.concat(songStr);
    }

    result = result.concat(Util.newLine);

    for (Enumeration e = artistCount.keys(); e.hasMoreElements() ;) {

        int artistID = ((Integer) e.nextElement()).intValue();

        String artistStr = artistCount.get(artistID)
                                + " songs are by the artist "
                                + artistNames.get(new
Integer(artistID))
                                + " (" + artistID + ") "
                                + Util.newLine;

        result = result.concat(artistStr);
    }

    result = result.concat(Util.newLine);

    for (Enumeration e = albumCount.keys(); e.hasMoreElements() ;) {

        int albumID = ((Integer) e.nextElement()).intValue();

        String albumStr = albumCount.get(albumID)
                                + " songs are from the album "
                                + albumNames.get(new
Integer(albumID))
                                + " (" + albumID + ") "
                                + Util.newLine;

        result = result.concat(albumStr);
    }

    result = result.concat(Util.newLine);

    for (Enumeration e = querySource.keys(); e.hasMoreElements() ;) {

        int source = ((Integer) e.nextElement()).intValue();
        int songCount = querySource.get(source);
        double doubleCount = new Double(songCount).doubleValue();

        String str = songCount

```



```

Integer(source))
        + " songs ("
        + ((doubleCount / length()) * 100)
        + "%) are from the "
        + querySourceName.get(new
        + " query"
        + Util.newLine;

        result = result.concat(str);
    }

    result = result.concat(Util.newLine);

    for (Enumeration e = originSource.keys(); e.hasMoreElements() ;) {

        int source = ((Integer) e.nextElement()).intValue();
        int songCount = originSource.get(source);
        double doubleCount = new Double(songCount).doubleValue();

        String str = songCount
        + " songs ("
        + ((doubleCount / length()) * 100)
        + "%) originated from "
        + originSourceName.get(new
        + Util.newLine;

        result = result.concat(str);
    }

    result = result.concat(Util.newLine);

    int bucketSize = 100 / BUCKET_COUNT;
    double playlistLength = media.size();

    for (int i = 0; i < BUCKET_COUNT; i++)
    {
        result = result.concat(
            "Percentile "
            + (i * bucketSize) + "% - "
            + ((i + 1) * bucketSize) + "%: " + buckets[i] +
            + Util.fix(100 * (buckets[i] / playlistLength),
            " ("
            2, 0) + "%)" + Util.newLine);
    }

    return (result + Util.newLine);
}

public int length ()
{
    return media.size();
}

public void append (SongData song)

```

```

        {
            float bucketSize = (new Float(101)).floatValue() / (new
Float(BUCKET_COUNT)).floatValue();
            int bucket = (int) Math.floor(song.status.percentile / bucketSize);
            // Util.debug("adding mediaID " + song.mediaID
            //          + " in percentile " + song.status.percentile + " (bucket
            //          + bucket + ")");
            media.addElement(song);
            buckets[bucket]++;
        }

    public Playlist shuffle()
    {
        Vector newList = new Vector(media.size());

        int rand = 0;

        while (media.size() > 0)
        {
            rand = (int) Util.random(media.size() - 1);

            Object m = media.elementAt(rand);
            media.removeElementAt(rand);
            newList.addElement(m);
        }

        media = newList;

        return this;
    }

    public int nextOrdinal(DBConnection conn)
    {
        int ordinal = 1;

        try
        {
            DBResultSet rs = conn.executeQuery("exec sp_lcGetOrdinalID_xsxx " +
userID);

            while (!rs.getBOF() && !rs.getEOF())
            {
                ordinal = rs.getInt("ordinal");

                rs.next();
            }

            conn.executeQuery("exec sp_lcUpdatePlaylistData_ixxd "
+ userID + ", ")

```

```

        + djID + ", "
        + moodID + ", "
        + mediaType);
    }
    catch (DBException oops)
    {
        Util.debug("DB Exception in Playlist::nextOrdinal: " +
oops.getMessage());
    }

    return ordinal;
}

public void deleteHighOrdinals(DBConnection conn, int ordinal)
{
    try
    {
        conn.executeUpdate("exec sp_lcDeletePlaylistRange_xxxd "
            + userID + ", "
            + ordinal);
    }
    catch (DBException oops)
    {
        Util.debug("DB Exception in Playlist::deleteHighOrdinals: " +
oops.getMessage());
    }
}

private SimplePlaylist toSimplePlaylist()
{
    SimplePlaylist result = new SimplePlaylist();

    result.mediaType = this.mediaType;
    result.djID      = this.djID;
    result.moodID     = this.moodID;

    // copy playlist
    for (int i = 0; i < media.size(); i++)
    {
        result.songs.addElement(((SongData)
media.elementAt(i)).toSimpleClip(mediaType));
    }

    // copy news
    for (int i = 0; i < news.size(); i++)
    {
        result.news.addElement(((Clip)
news.elementAt(i)).toSimpleClip(mediaType));
    }

    // copy ads

```

```

        for (int i = 0; i < ads.size(); i++)
        {
            result.ads.addElement(((Clip)
ads.elementAt(i)).toSimpleClip(mediaType));
        }

        // copy tips
        for (int i = 0; i < tips.size(); i++)
        {
            result.tips.addElement(((Clip)
tips.elementAt(i)).toSimpleClip(mediaType));
        }

        return result;
    }

    public void save (DBConnection conn, SimplePlaylist oldPlaylist)
    {
        Date startDate = new Date();

        SimplePlaylist thoreau = toSimplePlaylist();

        Util.printElapsedTime("Convert to SimplePlaylist", startDate);

        if (oldPlaylist != null)
        {
            thoreau.lastAd    = oldPlaylist.lastAd;
            thoreau.lastNews = oldPlaylist.lastNews;
            thoreau.lastTip   = oldPlaylist.lastTip;
        }

        thoreau.save(conn, userID);

        Util.printElapsedTime("SavePlaylist", startDate);
    }

    /*
    public boolean save (DBConnection conn)
    {

        if (length() <= 0)
            return false;

        boolean resetOrdinal = false;
        int highOrdinal, ordinal;
        Date startDate = new Date();

        highOrdinal = ordinal = nextOrdinal(conn);

        if (highOrdinal > MAX_ORDINAL)
        {
            ordinal = 1;
            resetOrdinal = true;
        }

        Util.printElapsedTime("GetOrdinal", startDate);
    }

```

```

        Thread saveNews = new SaveClips(news, "sp_lcSaveNewsPlaylist_ixxd",
ordinal, mediaType, userID);
        Thread saveAds = new SaveClips(ads, "sp_lcSaveAdsPlaylist_ixxd",
ordinal, mediaType, userID);
        Thread saveTips = new SaveClips(tips, "sp_lcSaveTipsPlaylist_ixxd",
ordinal, mediaType, userID);

        int partition = (int) Math.round(media.size() / 4.0);

        Thread savePlaylist1 = new SavePlaylist(this, 0, partition, ordinal);
        Thread savePlaylist2 = new SavePlaylist(this, partition, partition * 2,
ordinal + partition);
        Thread savePlaylist3 = new SavePlaylist(this, partition * 2, partition
* 3, ordinal + (partition * 2));
        Thread savePlaylist4 = new SavePlaylist(this, partition * 3,
media.size(), ordinal + (partition * 3));

        savePlaylist1.start();
        savePlaylist2.start();
        savePlaylist3.start();
        savePlaylist4.start();

        saveNews.start();
        saveAds.start();
        saveTips.start();

        deleteHighOrdinals(conn, highOrdinal - 1);

        // everybody done yet?

        saveOrigins(conn);

        try
        {
            saveNews.join();
            saveAds.join();
            saveTips.join();
            savePlaylist1.join();
            savePlaylist2.join();
            savePlaylist3.join();
            savePlaylist4.join();
        }
        catch (InterruptedException e)
        {
            Util.debug("Playlist::save was interrupted while waiting");
        }

        Util.printElapsedTime("SavePlaylist", startDate);

        return true;
    }
    */

    private void saveClips(DBConnection conn, Vector clips, String storedProc)
    {
        for (int i = 0; i < clips.size(); i++)
        {

```

```

        Clip aClip = (Clip) clips.elementAt(i);

        String sql = "exec " + storedProc + " "
            + ID + ", "
            + aClip.mediaID + ", "
            + mediaType + ", "
            + userID;

        try
        {
            DBResultSet rs = conn.executeSQL(sql);
        }
        catch (DBException oops)
        {
            Util.debug("DB Exception: " + oops.getMessage());
        }
    }

    public String newLine()
    {
        return Util.newLine;
    }

    public String toASX()
    {
        String asx = "<ASX VERSION=\"3.0\" PREVIEWMODE=\"NO\">" + Util.newLine
            + Util.tab() + "<REPEAT>" + Util.newLine;

        String streamURL = Constants.STREAM_URL + "?u="
            + userID;

        for (int i = 0; i < 10; i++)
        {
            asx = asx.concat(Util.tab(2) +
                "<ENTRY>" + Util.newLine
                + Util.tab(3)
                + "<REF HREF=\""
                + streamURL
                + "&n="
                + i
                + ".asp"
                + "\"/>" + Util.newLine
                + Util.tab(2)
                + "</ENTRY>" + Util.newLine);
        }

        asx = asx.concat(Util.tab() + "</REPEAT>" + Util.newLine
            + "</ASX>" + Util.newLine);

        return asx;
    }
}

```

```

package com.launch.PlaylistGenerator;

import java.util.*;

//-----
/**
 * @author Ted Leung
 * @version 1999-09-22
 */
//-----

public final class Playlist2 implements java.io.Serializable
{

    //*****
    // variables
    //*****

    /** all these vectors contain exclusively Strings which are
    directory/filename of audio files */
    public Vector songs;
    public Vector news;
    public Vector ads;
    public Vector tips;

    //*****
    // methods
    //*****

    public Playlist2()
    {
        songs = new Vector(50);
        news  = new Vector(10);
        ads   = new Vector(10);
        tips  = new Vector(10);
    }

    //-----
    /**
    **/
    //-----

    public final String toString()
    {
        return
        (
            "songs="+songs.toString() + ", " +
            "news="+news.toString() + ", " +
            "ads="+ads.toString() + ", " +
            "tips="+tips.toString()
        );
    }
}

```

```
} //*****
```

D:\My Documents\email\Launch\PlaylistGenerator\Playlist2.java Page 2 of 2 11/05/99 1:28 PM


```

package com.launch.PlaylistGenerator;

public class PlaylistCreatorTest
{
    public static void main(String[] args)
    {
        Util.debug("using database server " + Constants.DB_SERVER);

        SongInfoCache songCache = new SongInfoCache(null);
        songCache.ratingsCache = new RatingsCache();

        // PlaylistParameters params = new PlaylistParameters(3771, null, 0,
        13302);
        PlaylistParameters params = new PlaylistParameters(6474126, null, 0,
        6474126);
        PlaylistGenerator gen      = new PlaylistGenerator(params, songCache,
        null);
        Playlist playlist          = gen.create(true, null);

        gen.toMatrix(null, Util.DISPLAY_TEXT);

        System.exit(0);
    }
}
D:\My Documents\email\Launch\PlaylistGenerator\PlaylistCreatorTest.java  Page 1 of 1  11/05/99 1:35
PM

```

```
package com.launch.PlaylistGenerator;
```

```
import java.io.*;
```

```
public class PlaylistEntry implements Serializable  
{
```

```
    public String title, filepath, songTitle, albumTitle, artistTitle;  
    public int mediaID, songID, albumID, artistID;
```

```
    public short implicit;  
    public byte origin;
```

```
}
```

```
D:\My Documents\email\Launch\PlaylistGenerator\PlaylistEntry.java  Page 1 of 1  11/05/99 1:28 PM
```

```

package com.launch.PlaylistGenerator;

import java.util.Vector;
import java.util.Date;
import javax.servlet.ServletOutputStream;
import java.util.Enumeration;

```

```

→ public class PlaylistGenerator
{

    public final static byte RATER_DJ      = 1;
    public final static byte RATER_BDS     = 2;
    public final static byte RATER_GENRE   = 3;

    private short factor      = (short)Constants.DEFAULT_PICK_FACTOR;
    private short ratio       = (short) Constants.DEFAULT_UNRATED_RATIO;
    private int playlistSize  = Constants.DEFAULT_PLAYLIST_SIZE;
    private int playlistID;

    private boolean haveTitles = false;

    private Date startDate;
    private Date lastDate;

    private int userID;
    private int djID;
    private int moodID;
    private short mediaType;

    private IntHash ratings;
    private ItemsProfile items;
    private PlayDates lastPlayed;

    private Population songs;
    private Vector news;
    private Vector ads;
    private Vector tips;
    private DJList djs;
    private GenreList genres;

    private Bandwidth speed;
    private MediaFormat format;

    private StationList stations;

    private ServletOutputStream out;

    private SongInfoCache songCache;

    private boolean playExplicitLyrics = true;

    /**
     * Creates a new playlist generator.
     */
}

```

```

public PlaylistGenerator()
{
    songs      = new Population();
    news       = new Vector();
    ads        = new Vector();
    tips       = new Vector();
    ratings    = new IntHash();
    djs        = new DJList();
    items      = new ItemsProfile();
    lastPlayed = new PlayDates();
    genres     = new GenreList();
    stations   = new StationList();
}

public PlaylistGenerator (PlaylistParameters params, SongInfoCache cache,
ServletOutputStream out)
{
    this();

    userID      = params.userID;
    moodID      = params.moodID;
    djID        = params.djID;

    if (djID <= 0) djID = userID;

    speed       = params.speed;
    format      = params.format;
    playlistSize = params.playlistSize;
    songCache   = cache;
    this.out    = out;
}

private void getRandom()
{
    Date startDate = new Date();
    Song ditty;
    SongData data;
    SongInfo info;
    SongList songList;
    int rowCount = 0;
    double pickCount;
    double totalSongs;

    // the simple way

    /*
    songList = cache.getInGenres(genres);

    pickCount = Math.min(songList.size(), this.RANDOM_SONGS_COUNT);

    // import them all
    if (pickCount == songList.size())

```

```

    {
        for (int i = 0; i < pickCount; i++)
        {
            info = songList.elementAt(i);
            rowCount += addRandom(info, SongData.SOURCE_RANDOM);
        }
    }
    // import a random subset
    else
    {
        for (int i = 0; i < pickCount; i++)
        {
            info = songList.pickRandom();
            rowCount += addRandom(info, SongData.SOURCE_RANDOM);
        }
    }
    */

    // the faster(?) but way more complicated way

    int songCount = songCache.countInGenres(genres);
    totalSongs = songCache.size(SongInfoCache.TYPE_SONG);
    double percent = (songCount / totalSongs) * 100.0;

    Util.printElapsedTime("GetRandom done counting in genres", startDate);

    // the problem is if we pick randomly and they want songs from
    // only a few genres, we're probably not going to get enough to create
    // a playlist. So instead, if there's not a whole lot of songs in those
genres,
    // just get them directly from the genres instead of taking our chances
with random

    Util.debug("getRandom: " + songCount + " non-unique songs in genres ("
+ percent + "%)");

    if (percent < Constants.MIN_SONGS_IN_GENRES_TO_GET_RANDOM)
    {
        Util.debug("getRandom: getting directly from genres");

        // get the list of songs from each genre
        // choose the number to pick from each, proportional to the
number of songs
        // pick them

        int totalToPick = Math.min(Constants.RANDOM_SONGS_COUNT,
songCount);

        for (int i = 0; i < genres.size(); i++)
        {
            songList = songCache.getInGenre(genres.genreAt(i));
            pickCount = totalToPick * (songList.size() / ((double)
songCount));

            for (int j = 0; j < pickCount; j++)

```

```

        {
            info = songList.pickRandom();

            if (info != null)
            {
                rowCount += addRandom(info,
SongData.SOURCE_GENRES);
            }
        }
    }
    else
    {
        Util.debug("getRandom: picking randomly from all songs");

        for (int i = 0; i < Constants.RANDOM_SONGS_COUNT; i++)
        {
            // this is really fast
            info = songCache.randomSong();
            // this is really slow
            rowCount += addRandom(info, SongData.SOURCE_RANDOM);
        }
    }

    Util.debug("getRandom added " + rowCount + " songs");
    Util.printElapsedTime("GetRandom done", startDate);
}

private int addRandom(SongInfo info, byte source)
{
    SongData data = songs.initSongGetData(info.songID, Song.UNRATED);

    if (data != null)
    {
        data.querySource = source;
        data.setInfo(info);
        return 1;
    }

    return 0;
}

private void getPopular(SongList list)
{
    Date startDate = new Date();
    Song ditty;
    SongData data;
    SongInfo info;

    int rowCount = 0;

```

```

        if (list != null)
        {
            for (int i = 0; i < list.size(); i++)
            {
                info = list.elementAt(i);

                data = songs.getSongData(info.songID);

                if (data != null)
                {
                    // we can't add it, but let's append the info while
                    data.setInfo(info);
                }
                else
                {
                    data = songs.initSongGetData(info.songID,
Song.UNRATED);

                    if (data != null)
                    {
                        data.querySource = data.SOURCE_POPULAR;
                        data.setInfo(info);
                    }
                    rowCount++;
                }
            }

            Util.debug("getPopular added " + rowCount + " songs");
        }

/**
 * Gets all the required media and data to generate a playlist.
 */
→ private void gatherMedia(DBConnection conn)
{
    Thread getLastPlayed = new GetLastPlayed(lastPlayed, userID, out);

    Util.out(out, "starting gathering threads at " + timeStamp());

    // try to start them in ascending order of speed

    getLastPlayed.start();

    // get djs, genres, and bds subscriptions
    getSubscriptions(conn, djID, moodID);

    Util.out(out, "getSubscriptions done " + timeStamp());

    // we need to wait for the djs to come in first

```

```

→ Thread getRatings = new GetRatings(songs, items, djID, djs, songCache,
                                     out);                                (Exhibit 5, pp. 62 to 65)
getRatings.start();

Util.out(out, "All threads started " + timeStamp());

// getpopular and getrandom should not be threads since they are purely
processor bound now
getPopular(songCache.getPopular(mediaType));

Util.out(out, "getPopular done " + timeStamp());

getRandom();

Util.out(out, "getRandom done (picked " + Constants.RANDOM_SONGS_COUNT
+ " songs)" + timeStamp());

Util.out(out, "genres for mood " + moodID + ":" + genres.toString());

// wait for them to finish
try
{
    getRatings.join();
    getLastPlayed.join();
}
catch (InterruptedException oops)
{
    Util.debug("InterruptedException: " + oops.toString());
}

Util.out(out, "gatherMedia done " + timeStamp());
}

public void getSubscriptions(DBConnection conn, int userID, int moodID)
{
    Date started = new Date();

    try
    {
        DBResultSet rs = conn.executeQuery("exec
sp_lcoGetAllSubscriptions_xxxx "
                                           + userID + ", "
                                           + moodID);

        int raterID;
        int raterType;

        while (!rs.getBOF() && !rs.getEOF())
        {

```



```

        raterID    = rs.getInt("raterID");
        raterType  = rs.getInt("raterType");

        if (raterType == RATER_DJ)
        {
            djs.addElement(new DJ(raterID));
        }
        else if (raterType == RATER_GENRE)
        {
            genres.add((short) raterID);
        }
        else if (raterType == RATER_BDS)
        {
            stations.addElement(new Station(raterID));
        }

        rs.next();
    }

    Util.debug("getSubscriptions added "
               + djs.size() + " DJs, "
               + genres.size() + " Genres, "
               + stations.size() + " Stations");
}
catch (DBException oops)
{
    Util.debug("DB Exception in getSubscriptions " +
oops.getMessage());
}

Util.printElapsedTime("getSubscriptions took ", started);
}

/**
Calculates scores for all the songs and puts them into the various vectors
*/
public void processSongs()
{
    byte result;
    WeightMatrix weights = new WeightMatrix();

    Integer songID;
    Song aSong;
    SongData data;
    short type;
    Date playedAt;
    SongInfo info;
    int good = 0;
    int tested = 0;
    int artistID, albumID;
    Item albumItem;
    Item artistItem;

```

```

AlbumArtistData albumAndArtist = new AlbumArtistData();

IntHash reasons = new IntHash();

double now = lastPlayed.dbDate.getTime();
double lastThreeHours = Util.MILLISECONDS_IN_SECOND *
                        Util.SECONDS_IN_MINUTE *
                        Util.MINUTES_IN_HOUR *
                        3;

for (Enumeration e = songs.keys(); e.hasMoreElements() ;)
{
    tested++;

    albumAndArtist.reset();

    songID = (Integer) e.nextElement();
    aSong = songs.get(songID);
    data = aSong.getData();

    if (aSong.getType() == Song.EXCLUDED)
    {
        reasons.increment(1);
    }
    else
    {
        // add the song info

        info = data.getInfo();

        // get the song info from the cache
        if (info == null)
        {
            info = (SongInfo) songCache.get(songID,
SongInfoCache.TYPE_SONG);
            data.setInfo(info);
        }

        // if it's still null, it's not encoded
        if (info == null)
        {
            aSong.setType(Song.EXCLUDED);
            reasons.increment(2);
            continue;
        }

        // ok, we have the song info.

        // add last played

        playedAt = lastPlayed.get(songID);
    }
}

```

```

if (playedAt != null)
{
    lastPlayed.remove(songID);

    // don't play the same song twice in a 3 hour period
    if (now - playedAt.getTime() < lastThreeHours)
    {

        // mark songs played in the last three hours
        // so as to comply with the RIAA rules
        // and make sure we don't pick too many later

        artistID = data.getArtistID();
        albumID = data.getAlbumID();

        // "various artists" albums don't count
        if (!ArtistInfo.isVariousArtists(artistID))
        {
            songs.artistCounts.increment(artistID);
        }

        songs.albumCounts.increment(albumID);

        // make sure we don't play this again so soon
        aSong.setType(Song.EXCLUDED);
        reasons.increment(3);
        continue;
    }

    data.lastPlayed = lastPlayed.getScore(songID);
}

// check for bad words

if (!playExplicitLyrics && info.hasExplicitLyrics())
{
    aSong.setType(Song.EXCLUDED);
    reasons.increment(4);
    continue;
}

// now check for media in the type we need

if (!info.media.inType(mediaType))
{
    aSong.setType(Song.EXCLUDED);
    reasons.increment(5);
    continue;
}

// check for valid genres

if (!info.album.inGenres(genres))
{

```

rated the song

```
// for popular songs, don't exclude them,
// otherwise we won't be able to default to them
// if the genre restrictions are too tight

if (data.querySource == data.SOURCE_POPULAR)
{
    songs.remove(songID);
}

reasons.increment(6);
aSong.setType(Song.EXCLUDED);
continue;
}

// we got this far, so try to calculate an implicit rating
result = data.calculateImplicit(items, albumAndArtist);

if (result == SongData.EXCLUDE_ME)
{
    aSong.setType(Song.EXCLUDED);
    reasons.increment(7);
    continue;
}

if (result == SongData.MAKE_ME_IMPLICIT)
{
    aSong.setType(Song.IMPLICIT);
    data.calculateDJs(items, albumAndArtist);
    data.score(weights, stations);
    songs.implicit.addElement(data);
    good++;
}
else
{
    type = aSong.getType();

    // put the song in a list to pick from later

    if (type == Song.EXPLICIT)
    {
        // your djs don't matter if you explicitly
        songs.explicit.addElement(data);
    }
    else if (type == Song.IMPLICIT)
    {
        data.calculateDJs(items, albumAndArtist);
        songs.implicit.addElement(data);
    }
    else if (type == Song.UNRATED)
    {
        data.calculateDJs(items, albumAndArtist);
        songs.unrated.addElement(data);
    }
}
```

```

    }

    // calculate the score

    data.score(weights, stations);
    good++;
}
}

Util.out(out, "scores calculated " + timeStamp());

// for all the songs we didn't get for whatever reason, make sure we
// are accounting for their plays for compliance with RIAA rules
lastPlayed.markRecentlyPlayed(songCache, songs);

Util.out(out, "recently played albums and artists marked " +
timeStamp());

Util.out(out, "Of " + tested + " songs, these are the reasons for
exclusion: "

    + reasons.get(1) + " were already excluded, "
    + reasons.get(2) + " were not encoded, "
    + reasons.get(3) + " were played in the last 3 hours, "
    + reasons.get(4) + " had explicit lyrics, "
    + reasons.get(5) + " were not in mediaType " + mediaType +
", "

    + reasons.get(6) + " were not in their genres, "
    + reasons.get(7) + " had an implicit rating of 0.");

Util.out(out, "There are " + good + " songs available for play");
}

/**
 * Gets a user's preferences for their playlists
 */
public boolean getOptions(DBConnection conn)
{
    int rowCount = 0;
    short tempRatio;
    short bandwidth = 0;

    // returns: ratio, factor, mediaType

    String sql = "exec sp_lcGetPreferences_xsxx " + userID;

    try
    {
        DBResultSet rs = conn.executeQuery(sql);

        if (!rs.getBOF() && !rs.getEOF())
        {

```

```

        tempRatio = (short) rs.getInt("unratedQuota");

        if (tempRatio > 0 && tempRatio < 100)
            ratio = tempRatio;

        playExplicitLyrics = rs.getBoolean("explicit");

        // if there was no mediatype set from the parameters
        // set it to the default

        if (!speed.isSet())
            speed.set(rs.getShort("bandwidth"));

        rowCount++;
    }

    }
    catch (DBException oops)
    {
        Util.debug("DB Exception in getOptions: " + oops.getMessage());
    }

    mediaType = Media.getMediaType(speed, format);

    Util.debug("Play dirty songs?: " + playExplicitLyrics);
    Util.debug("Bandwidth: " + speed.toString());
    Util.debug("Format: " + format.toString());
    Util.debug("mediaType: " + mediaType);

    return (rowCount > 0);
}

/**
 * Creates a playlist.
 */

→ public Playlist createPlaylist(DBConnection conn)
{
    Util.out(out, "start of createPlaylist " + timeStamp());

    Playlist playlist = new Playlist(playlistID);

→    gatherMedia(conn);                                (Exhibit 5, p. 131)
→    processSongs();                                    (Exhibit 5, p. 133 to 137)

→    playlist = makePlaylist(factor, ratio, playlistSize, playlist);
                                                    (Exhibit 5, p. 140 to 141)

    Util.out(out, "end of createPlaylist " + timeStamp());

    return playlist;
}

private void logCreate(DBConnection conn)

```

```

{
    try
    {
        conn.executeSQL("exec sp_lcLogPlaylist_ixxx "
            + userID + ", "
            + djID + ", "
            + moodID + ", "
            + 0 + ", "
            + mediaType + ", "
            + elapsedTime()
            );
    }
    catch (DBException e)
    {
        Util.debug("DBException in logCreate: " + e.toString());
    }
}

/**
 * Creates and immediately saves a playlist.
 */
public Playlist create(boolean save, SimplePlaylist oldPlaylist)
{
    DBConnection conn = null;
    Playlist playlist = null;

    try
    {
        conn = new DBConnection();

        getOptions(conn);

        playlist = createPlaylist(conn);    (Exhibit 5, pp. 138)

        Util.out(out, "starting to save playlist " + timeStamp());

        if (save)
            playlist.save(conn, oldPlaylist);

        logCreate(conn);

        Util.out(out, "done saving playlist " + timeStamp());

        conn.close();
    }
    catch (DBException oops)
    {
        Util.out(out, "DBException in create: " + oops.getMessage());
    }
    catch (Throwable e)
    {
        System.err.println("Generic Exception caught in

```

```

PlaylistGenerator: " + e.toString());
    e.printStackTrace();
}

```

```

    return playlist;
}

```

```

→ public Playlist makePlaylist(int factor, int ratio, int playlistSize,
    Playlist playlist)
{

```

```

    Util.out(out, "ordering..." + timeStamp());

```

```

    songs.sort(songs.explicit);
    songs.sort(songs.implicit);
    songs.sort(songs.unrated);

```

```

    Util.out(out, "finished sorting vectors at " + timeStamp());

```

```

    playlist.counts = new PickCount(userID, djID, ratio, playlistSize,
songs, out);

```

```

    // set up the playlist

```

```

    playlist.userID    = this.userID;
    playlist.moodID    = this.moodID;
    playlist.djID      = this.djID;
    playlist.mediaType = this.mediaType;

```

```

    // copy the list of albums and artists recently played
    // for the RIAA rules

```

```

    playlist.albums = (IntHash) songs.albumCounts.clone();
    playlist.artists = (IntHash) songs.artistCounts.clone();

```

```

    // pick songs

```

```

→ pickSongs(playlist);

```

(Exhibit 5, pp. 141 to 143)

```

    // check if we got everything we need
    if (playlist.media.size() < playlistSize)
    {

```

```

        Util.out(out, "We only got " + playlist.media.size() + " songs
for user " + playlist.userID + ". Playing popular music in mediaType " +
mediaType);

```

```

        // uh oh, we didn't get enough songs; play popular stuff
        playlist.counts.explicit = 0;
        playlist.counts.implicit = 0;
        playlist.counts.unrated = playlistSize;

```

```

        playlist.albums = (IntHash) songs.albumCounts.clone();
        playlist.artists = (IntHash) songs.artistCounts.clone();

```

```

        playlist.resetSources();

```

```

        playlist.media.removeAllElements();

```



```

        playlist.popularOnly = true;

        songs.importPopular(songCache.getPopular(mediaType), lastPlayed,
playExplicitLyrics);

        pickSongs(playlist);
    }

    // pick news

    pickNews(playlist);

    Util.out(out, "picked news " + timeStamp());

    // pick ads

    pickAds(playlist);

    Util.out(out, "picked ads " + timeStamp());

    // pick tips

    pickTips(playlist);

    Util.out(out, "picked tips " + timeStamp());
    Util.out(out, "playlist has " + playlist.length() + " songs");
    Util.out(out, "shuffling playlist...");
    return playlist.shuffle();
}

public void pickNews(Playlist list)
{

    list.news = songCache.randomClipList(SongInfoCache.TYPE_NEWS,
mediaType, Constants.MAX_NEWS_ITEMS);

}

public void pickAds(Playlist list)
{

    list.ads = songCache.randomClipList(SongInfoCache.TYPE_AD, mediaType,
Constants.MAX_ADS);

}

public void pickTips(Playlist list)
{

    list.tips = songCache.randomClipList(SongInfoCache.TYPE_TIP, mediaType,
Constants.MAX_TIPS_ITEMS);

}

→ public Playlist pickSongs (Playlist list)
{

```

```

Util.out(out, "start of pickSongs " + timeStamp());

PickList pickTypes = new PickList(list.counts);

int pickOrder = 0;
int iteration = 0;

int artistID, albumID, artistCount, albumCount;

short type;
SongData pick;
SongGroup songGroup;

while (pickTypes.size() > 0)
{
    iteration++;
    pick = null;
    songGroup = null;

    // get a group to pick from
    type = pickTypes.getRandom();

    if (type == Song.EXPLICIT && songs.explicit.size() > 0)
    {
        songGroup = songs.explicit;
    }
    else if (type == Song.IMPLICIT && songs.implicit.size() > 0)
    {
        songGroup = songs.implicit;
    }
    else
    {
        type = Song.UNRATED;
        songGroup = songs.unrated;
    }

    // pick a random song from a group
    pick = songGroup.pickRandom(factor);

    // if we have none of that type, try another
    if (pick == null)
    {
        pickTypes.reAdd(type, songGroup, songs);
        continue;
    }

    artistID = pick.getArtistID();
    albumID = pick.getAlbumID();

    artistCount = 0;
    albumCount = 0;

    // check for RIAA compliance
    // various artists and soundtracks don't count

```

```

        if (!ArtistInfo.isVariousArtists(artistID))
            artistCount = list.artists.get(artistID);

        albumCount = list.albums.get(albumID);

        if (artistCount >= Constants.RIAA_MAX_SONGS_BY_ARTIST
            || albumCount >= Constants.RIAA_MAX_SONGS_FROM_ALBUM)
        {
            pick.status.status = PickStatus.REJECTED;
            // Util.debug("Song rejected by RIAA");

            // we have too many from this artist or album. Try again.
            pickTypes.reAdd(type, songGroup, songs);
            continue;
        }

        // increment the album and artist counts
        if (!ArtistInfo.isVariousArtists(artistID))
            list.artists.increment(artistID);

        list.albums.increment(albumID);

        // add it to the playlist
        list.append(pick);

        pick.status.status = PickStatus.PICKED;
        pick.status.order = ++pickOrder;
    }

    songs.ordered = false;

    Util.out(out, "end of pickSongs " + timeStamp());

    return list;
}

public void toMatrix(ServletOutputStream out, int displayType)
{
    songs.order();

    String h1begin = "";
    String h1end = "";

    if (displayType == Util.DISPLAY_HTML)
    {
        h1begin = "<P><H1>";
        h1end = "</H1>";
    }

    Util.out(out, h1begin + "Item Ratings" + h1end + Util.newLine);
}

```

```

        items.print(out, songCache);

        Util.out(out, h1begin + "Explicitly Rated Songs" + h1end +
Util.newLine);

        songs.toMatrix(out, Song.EXPLICIT, displayType);

        Util.out(out, h1begin + "Implicitly Rated Songs" + h1end +
Util.newLine);

        songs.toMatrix(out, Song.IMPLICIT, displayType);

        Util.out(out, h1begin + "Unrated Songs" + h1end + Util.newLine);

        songs.toMatrix(out, Song.UNRATED, displayType);

//          + h1begin + "Excluded Songs" + h1end + Util.newLine
//          + songs.excludedList();

    }

    public String timeStamp()
    {
        Date now = new Date();

        if (startDate == null)
        {
            startDate = lastDate = now;
        }

        double diff = (now.getTime() - lastDate.getTime()) / 1000.0;
        double total = (now.getTime() - startDate.getTime()) / 1000.0;
        lastDate = now;

        return Util.newLine
            + "-----" + Util.newLine
            + diff + " lap time, " + total + " total" + Util.newLine
            + "-----" + Util.newLine;
    }

    public double elapsedTime()
    {
        Date now = new Date();

        if (startDate == null)
        {
            startDate = lastDate = now;
        }

        return (now.getTime() - startDate.getTime()) / 1000.0;
    }
}

```

```

package com.launch.PlaylistGenerator;

import java.io.*;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import java.util.*;

/**
 *-----
 *
 * PlaylistGeneratorServlet.java 6/30/99
 * Servlet that creates LAUNCHcast playlists
 * Copyright (c) 1999 Launch, Inc.
 * @author Jeff Boulter
 *-----
 */
→ public class PlaylistGeneratorServlet extends HttpServlet {

    SongInfoCache songCache;
    Thread cacheUpdater;

    public void generatePlaylist(HttpServletRequest request,
        HttpServletResponse response) throws IOException
    {

        // get stream for output
        ServletOutputStream out = response.getOutputStream();

        GeneratorParameters prop = new GeneratorParameters(request);

        if (prop.debug())
            response.setContentType("text/plain");
        else
            response.setContentType("video/x-ms-asf");

        PlaylistParameters params = new PlaylistParameters(prop);
        PlaylistStatus status = new PlaylistStatus(prop.userID());
        status.init(out);
        if (prop.debug())
            out.print(status.toString());

        boolean generate = true;
        // no need to regenerate right now, use an old playlist
        if (prop.forceRefresh())
        {
            if (prop.debug()) out.println("generating because forceRefresh is on");
        }
        else if (status.isStale())
        {
            if (prop.debug()) out.println("generating because the playlist is stale");
        }
        else if (prop.speed().isSet() && (prop.speed().get() != status.speed.get()))
        {
            if (prop.debug()) out.println("generating because the mediaTypes are different");
        }
        else if (prop.format().isSet() && (prop.format().get() != status.format.get()))
        {
            if (prop.debug()) out.println("generating because the media formats are different");
        }
    }
}

```

```

    }
    else if (prop.moodID() != status.moodID)
    {
        if (prop.debug()) out.println("generating because the moods are different");
    }
    else if (prop.djID() != status.djID)
    {
        if (prop.debug()) out.println("generating because the djs are different");
    }
    else
        generate = false;
    if (!generate) // we can use an old playlist
    {
        // reset the ad, news, and tip dates

        if (status.playlist != null)
        {
            status.resetDates();
        }

        Playlist playlist = new Playlist();
        playlist.userID = status.userID;
        out.print(playlist.toASX());
    }
    else // we have to generate the playlist
    {
        ServletOutputStream outputStream = null;
        if (prop.debug())
        {
            outputStream = out;
            out.println("regenerating playlist with parameters: " + params.toString() +
"<PRE>");
            out.flush();
        }

        PlaylistGenerator gen = new PlaylistGenerator(params, songCache, outputStream); (Exhibit 5, p. 127)
        Playlist playlist = gen.create(!prop.dontsave(), null); (Exhibit 5, p. 139)

        if (prop.debug())
        {
            out.println("</PRE>");
            if (prop.debugFormat() == Util.DISPLAY_TEXT)
                out.println("<PRE>");
            out.println(playlist.toString()
                + "<P>");
            if (prop.matrix())
            {
                out.println("<FONT SIZE=-1>");
                gen.toMatrix(out, prop.debugFormat());
                out.println("</FONT>");
            }
            if (prop.debugFormat() == Util.DISPLAY_TEXT)
                out.println("</PRE>");
            out.println("<XMP>" + playlist.toASX() + "</XMP>");
        }
    }

```

```

        }
        else
            out.print(playlist.toASX());
    }

    out.close();
}

public void refreshPlaylist(HttpServletRequest request,
    HttpServletResponse response) throws IOException
{
    // get stream for output
    ServletOutputStream out = response.getOutputStream();

    response.setContentType("text/plain");
    // this is the stuff coming in on the query string
    GeneratorParameters prop = new GeneratorParameters(request);
    PlaylistParameters params = new PlaylistParameters(prop);

    // this is what's in their current playlist
    PlaylistStatus status = new PlaylistStatus(prop.userID());
    status.init(out);

    if (prop.debug())
        out.print(status.toString());

    if (status.isStale())
    {
        ServletOutputStream outStream = null;

        params = new PlaylistParameters(status);

        if (prop.debug())
        {
            outStream = out;
            out.println("refreshing playlist with parameters: " + params.toString());
            out.flush();
        }

        PlaylistGenerator gen = new PlaylistGenerator(params, songCache, outStream);
        Playlist playlist = gen.create(!prop.dontsave(), status.playlist);
    }
    else
    {
        out.println("No need to refresh playlist now");
    }

    out.close();
}

public void doGet (
    HttpServletRequest request,
    HttpServletResponse response
) throws ServletException, IOException {

    try
    {
        //Util.debug("PlaylistGeneratorServlet recieved a Get");
    }
}

```

```

// prevent caching
response.setHeader("Pragma", "no-cache");
response.setHeader("Cache-control", "no-cache");
response.setHeader("Expires", "0");

// figure out what we need to do

String actionStr = request.getParameter("action");
if (actionStr == null)
    actionStr = new String("generate");
if (actionStr.equals("refresh"))
{
    refreshPlaylist(request, response);
}
else if (actionStr.equals("cachestatus"))
{
    ServletOutputStream out = response.getOutputStream();
    response.setContentType("text/plain");
    songCache.ratingsCache.status(out, request.getParameter("detail") != null);

    out.close();
}
else //default action
{
    generatePlaylist(request, response);
}
}
catch (Throwable e)
{
    System.err.println(new Date().toString() + " Caught an exception in doGet: " +
e.toString());
    e.printStackTrace();
}
}

public void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException
{
    Util.debug("PlaylistGeneratorServlet recieved a Post");

    try
    {
        String user_agent=req.getHeader("USER_AGENT");

        if (user_agent.equals(com.launch.misc.constants.PLAYLIST_SERVER))
        {
            // need to generate play list and return it

            GeneratorParameters prop = new GeneratorParameters(req);
            PlaylistParameters params = new PlaylistParameters(prop);
            PlaylistGenerator gen = new PlaylistGenerator(params, songCache, null);
            Playlist playlist = gen.create(true, null);

            Playlist2 playlist2 = playlist.toPlaylist2();

            ObjectOutputStream oos=new ObjectOutputStream(resp.getOutputStream());
            oos.writeObject(playlist2);
            oos.flush();
            oos.close();
        }
        else if (user_agent.equals(com.launch.misc.constants.RATING_WIDGET))
    }
}

```



```

        {
            // need to update cache with new info

            int data_size=req.getContentLength();
            byte b[]=new byte[data_size];
            req.getInputStream().read(b,0,data_size);
            Vector v=(Vector)(new ObjectInputStream(new
ByteArrayInputStream(b))).readObject();

            Util.debug("received a list of changed ratings " + v);

            // need to tell cache of these changes
            Enumeration e=v.elements();
            while (e.hasMoreElements())
            {
                songCache.ratingsCache.putIntoCache((CachedRating)e.nextElement());
            }
        }
        else
        {
            System.err.println("PlaylistGeneratorServlet received a post from an unknown
person : " + user_agent);
        }
    }
    catch (Throwable t)
    {
        t.printStackTrace();
    }
}

/**
 * Initialization method -
 *
 */
public void init (ServletConfig config) throws ServletException
{
    super.init(config);
    songCache = new SongInfoCache(null);

    // start the updater thread
    cacheUpdater = new SongInfoCacheUpdater(this);
    cacheUpdater.setPriority(Thread.MIN_PRIORITY);
    cacheUpdater.start();

    songCache.ratingsCache = new RatingsCache();
}

/**
 * Destroy method -
 * get rid of the api
 * servlets "should have" a destroy method for garbage collection
 */
public void destroy()
{
    cacheUpdater.stop();
    cacheUpdater = null;
    songCache = null;
}
}

```

```

package com.launch.PlaylistGenerator;

import javax.servlet.ServletOutputStream;

/**
 * this is the dumb class for ASP
 */
public class PlaylistMaker
{

    public PlaylistGenerator generator;
    public Playlist playlist;

    public PlaylistMaker()
    {
        generator = new PlaylistGenerator();
    }

    public void init(int userID, int djID, short mediaType, int moodID, int
playlistID)
    {
        // generator.init(userID, djID, moodID);
    }

    public int make()
    {
        playlist = generator.create(false, null);

        return playlist.ID;
    }

    public int makeAndSave()
    {
        playlist = generator.create(true, null);
        return playlist.ID;
    }

    public void toMatrix(ServletOutputStream out, int displayType)
    {
        generator.toMatrix(out, displayType);
    }

    public String toASX()
    {
        return playlist.toASX();
    }

}

```

D:\My Documents\email\Launch\PlaylistGenerator\PlaylistMaker.java Page 1 of 1 11/05/99 1:32 PM

```

package com.launch.PlaylistGenerator;

public class PlaylistParameters
{
    int userID;
    int djID;
    int playlistSize = Constants.DEFAULT_PLAYLIST_SIZE;
    int moodID;

    Bandwidth speed = new Bandwidth();
    MediaFormat format = new MediaFormat();

    public PlaylistParameters(int userID)
    {
        this.userID = djID = userID;
    }

    public PlaylistParameters(int userID, Bandwidth speed, int moodID)
    {
        this(userID);

        if (speed != null)
        {
            this.speed = speed;
        }

        this.moodID = moodID;
    }

    public PlaylistParameters(int userID, Bandwidth speed, int moodID, int djID)
    {
        this(userID, speed, moodID);

        if (djID > 0)
            this.djID = djID;
    }

    public PlaylistParameters(PlaylistStatus status)
    {
        this(status.userID, status.speed, status.moodID, status.djID);
    }

    public PlaylistParameters(GeneratorParameters prop)
    {
        this(prop.userID(), prop.speed(), prop.moodID(), prop.djID());
    }

    public String toString()
    {
        return "userID=" + userID + ", "
            + "bandwidth=" + speed.toString() + ", "
            + "moodID=" + moodID + ", "
            + "djID=" + djID;
    }
}

```

}
D:\My Documents\email\Launch\PlaylistGenerator\PlaylistParameters.java
PM

Page 2 of 2 11/05/99 1:35

```

package com.launch.PlaylistGenerator;

import java.util.Date;
import javax.servlet.ServletOutputStream;

public class PlaylistStatus
{
    int userID, newRatingsCount, moodID, djID, songsRemaining;
    short mediaType;

    Date lastPlaylist = new Date();

    MediaFormat format;
    Bandwidth speed;

    Date dbDate = new Date();

    public SimplePlaylist playlist;

    public PlaylistStatus(int userID)
    {
        format = new MediaFormat(MediaFormat.WINDOWSMEDIA);
        this.userID = userID;
    }

    public String toString()
    {
        return "Playlist status for userID " + userID + ":" + Util.newLine
            + " newRatingsCount: " + newRatingsCount + Util.newLine
            + " moodID: " + moodID + Util.newLine
            + " djID: " + djID + Util.newLine
            + " songsRemaining: " + songsRemaining + Util.newLine
            + " mediaType: " + mediaType + Util.newLine;
    }

    public void init(ServletOutputStream out)
    {
        try
        {
            DBConnection conn = new DBConnection();

            DBResultSet rs = conn.executeQuery("exec
sp_lcGetPlaylistInfoForUser_xsxx " + userID);

            while (!rs.getBOF() && !rs.getEOF())
            {
                newRatingsCount = rs.getInt("newRatingsCount");
                lastPlaylist      = rs.getTimestamp("lastPlaylist");
                dbDate            = rs.getTimestamp("dbDate");
                playlist          =
SimplePlaylist.fromBytes(rs.getBytes("playlist"));
                rs.next();
            }

            if (playlist != null)

```

```

        {
            songsRemaining = playlist.songs.size();
            moodID          = playlist.moodID;
            djID            = playlist.djID;
            mediaType       = playlist.mediaType;
            speed           = Media.typeToBandwidth(mediaType);
        }

        conn.close();
    }
    catch (DBException oops)
    {
        Util.out(out, "DBException in PlaylistStatus.init: " +
oops.toString());
    }

}

public void resetDates()
{
    if (playlist == null)
        return;

    Util.debug(new Date().toString() + " Playlist OK, just resetting dates
for userID " + userID);
    playlist.resetDates(dbDate);
    playlist.save(userID);
}

public boolean isStale()
{
    double oneWeek = Util.MILLISECONDS_IN_SECOND *
                        Util.SECONDS_IN_MINUTE *
                        Util.MINUTES_IN_HOUR *
                        Util.HOURS_IN_DAY *
                        Util.DAYS_IN_WEEK;

    if (songsRemaining <= Constants.REFRESH_AT_SONGS_LEFT)
        return true;

    // if you're listening to someone else's station, your new ratings
    // won't make a difference
    if (newRatingsCount >= Constants.REFRESH_AT_NEW_RATINGS_COUNT && userID
== djID)
        return true;

    if (new Date().getTime() - lastPlaylist.getTime() > oneWeek)
        return true;

    return false;
}

/*
public void flushPlaylist(ServletOutputStream out)

```

```

    {
        try
        {
            DBConnection conn = new DBConnection();
            DBResultSet rs = conn.executeQuery("exec sp_lcFlushPlaylist_xxud "
+ userID);
            conn.close();
        }
        catch (DBException oops)
        {
            Util.out(out, "DBException in PlaylistStatus::flushPlaylist: " +
oops.toString());
        }
    }

    public void deletePlaylist(ServletOutputStream out)
    {
        try
        {
            DBConnection conn = new DBConnection();
            DBResultSet rs = conn.executeQuery("exec sp_lcDeletePlaylist_xxud "
+ userID);
            conn.close();
        }
        catch (DBException oops)
        {
            Util.out(out, "DBException in PlaylistStatus::deletePlaylist: " +
oops.toString());
        }
    }

    public void resetClipSchedule()
    {
        try
        {
            DBConnection conn = new DBConnection();
            DBResultSet rs = conn.executeQuery("exec
sp_lcResetClipSchedule_xxux " + userID);
            conn.close();
        }
        catch (DBException oops)
        {
            Util.debug("DBException in PlaylistStatus::resetDates: " +
oops.toString());
        }
    }
    */
}

```

D:\My Documents\email\Launch\PlaylistGenerator\PlaylistStatus.java Page 3 of 3 11/05/99 1:24 PM

```

package com.launch.PlaylistGenerator;

import java.util.Vector;
import java.util.Hashtable;
import java.util.Enumeration;

public class PopularSongs
{
    private Hashtable byMedia;

    public SongList get(short mediaType)
    {
        return (SongList) byMedia.get(new Short(mediaType));
    }

    public PopularSongs(Hashtable songs, Hashtable mediaTypes)
    {
        byMedia = new Hashtable(1);

        // make a list of all songs and sort them
        SongList all = new SongList(songs);
        all.sort();

        // create each of the song lists
        for (Enumeration e = mediaTypes.keys(); e.hasMoreElements();)
        {
            Short mediaType = new Short(((Integer)
e.nextElement()).shortValue());
            byMedia.put(mediaType, new SongList());
        }

        SongInfo info;
        Media track;
        SongList list;

        // put each into a separate list for each mediaType
        for (int i = 0; i < all.size(); i++)
        {
            info = all.elementAt(i);

            for (int j = 0; j < info.media.size(); j++)
            {
                track = info.media.typeAt(j);
                list = ((SongList) byMedia.get(new
Short(track.mediaType)));
                list.addElement(info);
            }
        }

        // truncate each list to the top 1000 most popular songs
        for (Enumeration e = mediaTypes.keys(); e.hasMoreElements();)
        {
            Short mediaType = new Short(((Integer)
e.nextElement()).shortValue());

```



```
list = (SongList) byMedia.get(mediaType);  
list.setSize(1000);
```

```
}
```

```
}
```

```
}
```

D:\My Documents\email\Launch\PlaylistGenerator\PopularSongs.java Page 2 of 2 11/05/99 1:24 PM

```

package com.launch.PlaylistGenerator;

import java.util.Enumeration;
import java.util.Date;
import java.text.SimpleDateFormat;
import java.util.Vector;
import java.util.Hashtable;
import javax.servlet.ServletOutputStream;
import java.text.DateFormat;

public class Population
{
    /*
    private int readers = 0;
    private int writersWaiting = 0;
    private boolean writing = false;
    */

    private boolean haveTitles = false;
    public boolean ordered = false;

    public SongGroup explicit;
    public SongGroup implicit;
    public SongGroup unrated;

    private Hashtable hash;

    public IntHash artistCounts;
    public IntHash albumCounts;

    public Population()
    {
        explicit      = new SongGroup();
        implicit      = new SongGroup();
        unrated       = new SongGroup();
        artistCounts  = new IntHash();
        albumCounts   = new IntHash();
        hash          = new Hashtable();
    }

    /*

    public synchronized void addReader()
    {
        ++readers;
    }

    public synchronized void removeReader()
    {
        --readers;
        if (readers == 0)
        {
            notifyAll();
        }
    }
    */

```

```

    }

    public synchronized void requestWrite()
    {
        ++writersWaiting;
    }

    public synchronized void finishWrite()
    {
        --writersWaiting;
        if (writersWaiting == 0)
        {
            notifyAll();
        }
    }

    */

    // returns this song if it's valid for adding data, null otherwise
    public synchronized Song initSong(int songID, short type)
    {
        if (type <= 0)
            return null;

        boolean result = true;

        /*
        requestWrite();

        while (readers > 0)
        {
            try { wait(); }
            catch (InterruptedException e) {}
        }

        writing = true;
        */

        Song song = get(songID);

        if (song == null)
        {
            song = new Song(songID, type);
            put(songID, song);

            // if it's excluded, it's not valid for modifying
            if (type == Song.EXCLUDED)
                result = false;
        }
        else
        {
            result = song.setType(type);
        }
    }

```

```

        if (result)
            return song;

//      writing = false;
//      finishWrite();

        return null;
    }

    public synchronized SongData initSongGetData(int songID, short type)
    {
        Song aSong = initSong(songID, type);

        if (aSong == null)
            return null;

        return aSong.getData();
    }

    public synchronized SongData getSongData(int songID)
    {
        return getSongData(new Integer(songID));
    }

    public synchronized SongData getSongData(Integer songID)
    {
        Song s = get(songID);

        if (s == null)
            return null;

        return s.getData();
    }

    public synchronized SongData getSongData(int songID, short type)
    {
        SongData result = null;

        /*
        synchronized (this)
        {
            while (writersWaiting > 0)
            {
                try { wait(); }
                catch (InterruptedException e) { }
            }
            addReader();
        }
        */

        Song song = get(songID);

        // there's no song for that ID; Did you call initSong?
        if (song != null && type >= song.getType())

```

```

        result = song.getData();
//    removeReader();
        return result;
    }

    public synchronized Song get(int songID)
    {
        return get(new Integer(songID));
    }

    public synchronized Song get(Integer songID)
    {
        return (Song) hash.get(songID);
    }

    public synchronized Song remove(int songID)
    {
        return remove(new Integer(songID));
    }

    public synchronized Song remove(Integer songID)
    {
        return (Song) hash.remove(songID);
    }

    private synchronized Song put(int songID, Song song)
    {
        return (Song) hash.put(new Integer(songID), song);
    }

    private int available()
    {
        int i = 0;

        for (Enumeration e = hash.keys(); e.hasMoreElements(); ) {
            Song song = get((Integer) e.nextElement());

            if (song.type != Song.EXCLUDED)
            {
                i++;
            }
        }

        return i;
    }

    public Enumeration keys()
    {
        return hash.keys();
    }

    public void order()

```

```

    {
        createVectors();
        sortVectors();
    }

    public int excludedCount()
    {
        int result = 0;

        for (Enumeration e = hash.keys(); e.hasMoreElements(); ) {
            Song song = get(((Integer) e.nextElement()).intValue());
            if (song.type == Song.EXCLUDED)
            {
                result++;
            }
        }

        return result;
    }

    public boolean isEligible(int songID, int artistID, int albumID)
    {
        Song song = get(songID);

        if (song != null && song.type == Song.EXCLUDED)
            return false;

        if ((artistCounts.get(artistID) < Constants.RIAA_MAX_SONGS_BY_ARTIST)
            && (albumCounts.get(albumID) <
Constants.RIAA_MAX_SONGS_FROM_ALBUM))
            return true;

        return false;
    }

    public void createVectors()
    {
        explicit.removeAllElements();
        implicit.removeAllElements();
        unrated.removeAllElements();

        for (Enumeration e = hash.keys(); e.hasMoreElements(); ) {
            // Util.debug("iteration " + i);
            Song mySong = get((Integer) e.nextElement());

            if (mySong != null)
            {
                SongData data = mySong.getData();

                if (mySong.type == Song.EXPLICIT)
                    explicit.addElement(data);
                else if (mySong.type == Song.IMPLICIT)

```

```

        implicit.addElement(data);
    else if (mySong.type != Song.EXCLUDED)
        unrated.addElement(data);
    }
}

public void importPopular(SongList abunch, PlayDates lastPlayed, boolean
playBadWords)
{
    SongInfo info;
    SongData data;
    Song ditty;
    int added = 0;

    unrated.setSize(0);

    long now = new Date().getTime();

    long lastThreeHours = Util.MILLISECONDS_IN_SECOND *
                           Util.SECONDS_IN_MINUTE *
                           Util.MINUTES_IN_HOUR *
                           3;

    long playedTime = 0;

    Date playedAt;

    for (int i = 0; i < abunch.size(); i++)
    {
        info = abunch.elementAt(i);
        playedAt = lastPlayed.get(info.songID);

        // don't play songs twice within 3 hours
        if (playedAt == null || (now - playedAt.getTime()) >
lastThreeHours)
        {
            if (playBadWords || !info.hasExplicitLyrics())
            {
                data = initSongGetData(info.songID, Song.UNRATED);

                if (data != null)
                {
                    data.setInfo(info);
                    unrated.addElement(data);
                    added++;
                }
            }
        }
    }

    Util.debug("import popular added " + added + " songs");
}

```

```

    }

    public void sortVectors()
    {
        sort(explicit, 0, explicit.size() - 1);
        sort(implicit, 0, implicit.size() - 1);
        sort(unrated, 0, unrated.size() - 1);

        // Util.debug("after sorting, ratedVector is: " + ratedVector.toString());
        // Util.debug("after sorting, unratedVector is: " +
        unratedVector.toString());

        ordered = true;
    }

    public void sort(Vector a)
    {
        sort(a, 0, a.size() - 1);
    }

    private void sort(Vector a, int from, int to)
    {
        // quicksort

        // If there is nothing to sort, return
        if ((a == null) || (a.size() < 2)) return;

        int i = from, j = to;
        SongData center = (SongData) a.elementAt((from + to) / 2);

        do {
            while((i < to) && (center.score < ((SongData)
a.elementAt(i)).score)) i++;
            while((j > from) && (center.score > ((SongData)
a.elementAt(j)).score)) j--;

            if (i < j) {
                SongData temp = (SongData) a.elementAt(i);
                a.setElementAt(a.elementAt(j), i);
                a.setElementAt(temp, j); // swap elements
            }

            if (i <= j) { i++; j--; }
        } while(i <= j);

        if (from < j) sort(a, from, j); // recursively sort the rest
        if (i < to) sort(a, i, to);

    }

    public String toString()
    {

```



```

String result = "";

for (Enumeration e = hash.keys(); e.hasMoreElements() ;) {

    int songID = ((Integer) e.nextElement()).intValue();
    Song song = get(songID);

    result = result.concat("songID " + songID
                           + " = " + song.toString()
                           + Util.newLine());
}

return result;
}

public String sourceCount()
{
    IntHash counts = new IntHash();
    String explicitList = "";

    for (Enumeration e = hash.keys(); e.hasMoreElements() ;) {

        Song song = get(((Integer) e.nextElement()).intValue());

        if (song.getType() == Song.EXPLICIT)
        {
            explicitList = explicitList.concat(song.songID + ", ");
        }

        counts.increment(song.type);
    }

    return "counts: EXPLICIT = " + counts.get(Song.EXPLICIT)
          + " (" + explicitList + ") "
          + " IMPLICIT = " + counts.get(Song.IMPLICIT)
          + " EXCLUDED = " + counts.get(Song.EXCLUDED);
}

public void toMatrix(ServletOutputStream out, int songType, int displayType)
{
    String delim = "";
    String prefix = "";
    String suffix = "";
    String rowPrefix = "";
    String rowSuffix = "";
    String bold = "";
    String unbold = "";

    if (displayType == Util.DISPLAY_HTML)

```

```

    {
        delim = "</TD><TD>";
        prefix = "<TABLE CELLPADDING=1 CELLSPACING=0>";
        suffix = "</TABLE>";
        rowPrefix = "<TR><TD>";
        rowSuffix = "</TD></TR>";
        bold = "<B><FONT SIZE=\"-1\">";
        unbold = "</FONT></B>";
    }
    else
    {
        delim = "\t";
    }

    Util.out(out, prefix);

    String header = Util.newLine + rowPrefix + bold
                    + Util.join(unbold + delim + bold,
SongData.namesArray())
                    + unbold + rowSuffix;

    Vector v = null;

    if (songType == Song.EXPLICIT)
        v = explicit;
    else if (songType == Song.IMPLICIT)
        v = implicit;
    else
        v = unrated;

    if (v != null)
    {
        for (int i = 0; i < v.size(); i++) {
            SongData data = (SongData) v.elementAt(i);

            if (i % 40 == 0)
                Util.out(out, header);

            Util.out(out, data.toString(displayType, (i + 1)));
        }
    }

    Util.out(out, suffix);
}

```

```

package com.launch.PlaylistGenerator;

public class Rating
{
    protected short rating;
    protected boolean set = false;

    public Rating()
    {

    }

    /**
     * create one with a default
     */

    public Rating(short defaultRating)
    {
        rating = defaultRating;
    }

    public boolean isSet()
    {
        return set;
    }

    public void set(short newRating)
    {
        rating = newRating;
        set = true;
    }

    public short get()
    {
        return rating;
    }

    public String toString()
    {
        if (!set)
            return rating + "(Not Set)";
        else
            return "" + rating;
    }
}

```

D:\My Documents\email\Launch\PlaylistGenerator\Rating.java Page 1 of 1 11/05/99 1:28 PM

```

package com.launch.PlaylistGenerator;

import java.util.*;
import javax.servlet.ServletOutputStream;
import java.io.IOException;

public final class RatingsCache implements GetRatingsCacheUsersInterface, Constants
{
    /**
     * This Hashtable will be of the form
     * (Integer userID, Hashtable CachedRating objects), if the Data in
     * the cache is invalid the entry will be of the form
     * (Integer userID, INVALID_DATA)
     * <br>
     * The Hashtable of CachedRating objects is of the form (Integer
itemID, CachedRating)
    **/

    private Hashtable ratingsList = new Hashtable(1);

    private GetRatingsCacheUsers gtu;

    private FrequencyCounter freq_counter = new
FrequencyCounter(RATINGS_CACHE_INITIAL_SIZE);

    private Date lastUpdated = new Date();
    private Date lastReset = new Date();

    //-----

    public RatingsCache()
    {
        gtu = new GetRatingsCacheUsers(this);

        // the following line is for testing purposes only, rem it out
otherwise.
        //
        gtu.SLEEP_TIME=5*60*1000;

        gtu.start();
    }

    /**
     * This method will get a list of rating for the given userids
     * @param userid an array of ints representing userids, each entry
should be a valid userID, do not pad with zeros.
     * @return a Vector of CachedRating objects
    **/
    public final Vector getRatings(Vector users)
    {
        //-----
        // algorithm
        //-----
        // check for userid in hashtable

```

```

// if found add to vector of ratings
// else build list of unfound things
//      get list of unfound things from database

Vector allRatings      = new Vector();
Integer userID;
Hashtable ratingProfile;
Vector nonCachedUsers = new Vector(users.size());
Date startDate         = new Date();
Enumeration e          = users.elements();

while (e.hasMoreElements())
{
    userID      = (Integer) e.nextElement();
    ratingProfile = (Hashtable) ratingsList.get(userID);

    if (ratingProfile == null)
    {
        Util.debug("RatingsCache MISS on user " + userID);
        nonCachedUsers.addElement(userID);
    }
    else
    {
        benchmark_date1 = new Date();

        Util.debug("RatingsCache HIT on user " + userID);
        appendToVector(allRatings, ratingProfile.elements());

        //      Util.printElapsedTime("Get from cache, " +
temp_hash.size() + " entries", benchmark_date1);
    }

    freq_counter.incrementValue(userID);
}

if (nonCachedUsers.size() > 0)
{
    MergeVectors(allRatings,
getRatingsFromDatabase(nonCachedUsers));
}

Util.printElapsedTime(Thread.currentThread().getName() + ", got "
+ allRatings.size() + " ratings ", startDate);
return allRatings;
}

public final void updateCachedUsers(Vector v)
{
    setCachedUserIDs(v);
}

public Hashtable getMostFrequentlyUsedUsers(int i)

```

```

{
    Hashtable h = freq_counter.getLargest(i);
    Vector    v = new Vector(h.size());

    // when we do this, also refresh the cache
    // to clean out any lingering data corruption

    Util.debug(new Date().toString() + " Resetting ratings cache");

    // clear the users in the cache
    setCachedUserIDs(v);

    lastReset = new Date();

    // put user hash into vector
    appendToVector(v, h.keys());

    // get all the ratings
    setCachedUserIDs(v);

    return h;
}

/**
 *
 */
public final void setCachedUserIDs(Vector userIDs)
{
    lastUpdated = new Date();

    Vector cachedUsers = (Vector) userIDs.clone();
    Date benchmark_date = new Date();

    if (cachedUsers.size() <= 0)
    {
        ratingsList = new Hashtable(1);
        Util.debug("setCachedUserIDs: no users passed");
        return;
    }

    Enumeration e = ratingsList.keys();
    Integer userID;

    // find the differences between the users already in the cache
    // and the new list of users
    // leave that result in cachedUsers

    // iterate through each user in the current cache
    while (e.hasMoreElements())
    {
        userID = (Integer) e.nextElement();

        // are they in the new list?
        if (cachedUsers.contains(userID))
        {

```

```

        // cool, just remove them from the new list
        cachedUsers.removeElement(userID);
    }
    else
    {
        // they've been removed
        ratingsList.remove(userID);
    }
}

Vector newRatings = new Vector();

// get all the ratings for the new cached users
if (cachedUsers.size() > 0)
{
    newRatings = getRatingsFromDatabase(cachedUsers);
    e = newRatings.elements();

    while (e.hasMoreElements())
    {
        putIntoCache((CachedRating) e.nextElement());
    }
}
else
{
    Util.debug(new Date().toString() + " setCachedUserIDs: no
new users in cache");
}

Util.printElapsedTime("refreshed cached users and loaded " +
newRatings.size() + " entries", benchmark_date);
}

/**
 *
 **/
private final Vector getRatingsFromDatabase(Vector userIDs)
{
    //-----
    // algorithm
    //-----
    // query database for info
    // build vector from resultsets.

    Vector results = new Vector(RATINGS_CACHE_INITIAL_SIZE);
    Date benchmark_date = new Date();

    //--- get item rating ---

    GetItemRatingsFromDB itemRatings = new
GetItemRatingsFromDB(userIDs, results);

    //--- get song rating ---

```

```

        GetSongRatingsFromDB songRatings = new
GetSongRatingsFromDB(userIDs, results);
        songRatings.start();
        itemRatings.start();

        //--- must wait for the two threads to finish ---

        try
        {
            itemRatings.join();
            songRatings.join();
        }
        catch (InterruptedException e)
        {
            System.err.println("PlaylistCache: interrupted
waiting for ratings, I'm not cleanning up...");
        }

        //--- done getting just return values ---

        Util.printElapsedTime("GetRatingsFromDatabase, " +
results.size() + " entries", benchmark_date);

        return results;
    }

    /**
     * appends the contents of vector2 into vector1
     */
    private static final void MergeVectors(Vector vector1, Vector vector2)
    {
        vector1.ensureCapacity(vector1.size() + vector2.size());

        Enumeration e = vector2.elements();

        while (e.hasMoreElements())
        {
            vector1.addElement(e.nextElement());
        }
    }

    public static final void appendToVector(Vector v, Enumeration e)
    {
        while (e.hasMoreElements())
        {
            v.addElement(e.nextElement());
        }
    }

    public static final String GetVectorAsCommaDelimitedList(Vector v)
    {
        if (v==null) return("");

```



```

String s=v.toString();
int vector_length=s.length();

if (vector_length >= 3)
{
    return(s.substring(1,vector_length-1));
}
else
{
    return("");
}
}

/**
 * This method adds the value to the hashtable pointed to by the key,
if the key does not exist yet it will create the first entry and the Hashtable
**/

public final void putIntoCache(CachedRating value)
{
    RatingsProfile profile = null;
    Integer userID = new Integer(value.userID);

    // this could be more efficient if we inserted all the ratings
for a particular user all at once
    if (ratingsList.containsKey(userID))
    {
        profile = (RatingsProfile) ratingsList.get(userID);
    }
    else
    {
        profile = new RatingsProfile(RATINGS_CACHE_INITIAL_SIZE);
        ratingsList.put(userID, profile);
    }

    if (value.rating < 0)
    {
        // unrate
        profile.remove(value.hashKey());
    }
    else
    {
        profile.put(value.hashKey(), value);
    }
}

public final String toString()
{
    return ratingsList.toString();
}

public final String userList()
{
    String result = "";

    Enumeration e = ratingsList.keys();
    Integer userID;

```

```

        while (e.hasMoreElements())
        {
            userID = (Integer) e.nextElement();
            result = result.concat(userID + ", ");
        }

        return result;
    }

    public final void status(ServletOutputStream out, boolean detail)
    throws IOException
    {
        out.print("RatingsCache has " + ratingsList.size() + " users" +
        Util.newLine
            + "Last Updated at " +
        lastUpdated.toString() + Util.newLine
            + "Last Reset at " +
        lastReset.toString() + Util.newLine
            + "UserList is " + userList() +
        Util.newLine);

        Enumeration e = ratingsList.keys();
        Integer userID;
        RatingsProfile profile;

        while (e.hasMoreElements())
        {
            userID = (Integer) e.nextElement();
            out.print(Util.newLine + "Profile for userID " + userID +
            ":" + Util.newLine);

            profile = (RatingsProfile) ratingsList.get(userID);

            if (profile == null)
            {
                out.print("NULL!" + Util.newLine);
            }
            else
            {
                out.print(Util.newLine +
            profile.count(Constants.ITEM_TYPE_SONG) + " songs, "
                +
            profile.count(Constants.ITEM_TYPE_ALBUM) + " albums, "
                +
            profile.count(Constants.ITEM_TYPE_ARTIST) + " artists, "
                +
            profile.count((byte) 0) + " total" + Util.newLine);

                if (detail)
                    out.print(profile.toString());
            }
        }
    }
}

```

```

package com.launch.PlaylistGenerator;

import java.util.Hashtable;
import java.util.Enumeration;

public class RatingsProfile extends Hashtable
{
    public RatingsProfile(int capacity)
    {
        super(capacity);
    }

    public int count(byte type)
    {
        int count = 0;

        if (type <= 0)
            return size();
        else
        {
            Enumeration e = keys();

            String key;

            CachedRating rating;

            while (e.hasMoreElements())
            {
                key = (String) e.nextElement();

                rating = get(key);

                if (rating.type == type)
                    count++;
            }

            return count;
        }
    }

    public CachedRating get(String key)
    {
        return (CachedRating) super.get(key);
    }

    public String toString()
    {
        String result = "";
        Enumeration e = keys();

        while (e.hasMoreElements())
        {
            result = result.concat((get((String)
e.nextElement()))).toString());
        }
    }
}

```

```
        return result;
```

```
    }
```

```
}
```

D:\My Documents\email\Launch\PlaylistGenerator\RatingsProfile.java Page 2 of 2 11/05/99 1:35 PM

```
package com.launch.PlaylistGenerator;
```

```
import java.util.*;
import java.io.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;
```

```
/**
```

```
*-----
```

```
*
```

```
* RatingWidgetServlet.java 7/8/99
* Initial Servlet for ratings Widget
* Copyright (c) 1999 LAUNCH Media, Inc.
* @author Jon Heiner
```

```
*-----
```

```
*/
```

```
→ public class RatingWidgetServlet extends HttpServlet implements
GetRatingsCacheUsersInterface, GetPlaylistServersInterface, Runnable
```

```
{
```

```
    private Vector cachedUsers = new Vector(1);
    private GetRatingsCacheUsers gtu;
    private Vector playlistServers = new Vector(1);
    private GetPlaylistServers gps;
```

```
    /** This vector contains CachedRating objects */
```

```
    private Vector dirtyRatings = new
Vector(Constants.RATING_UPDATE_LIST_INITIAL_SIZE);
    private Thread myThread;
```

```
    //-----
```

```
    /**
```

```
    * Handle requests...
```

```
    */
```

```
    public void doGet (
        HttpServletRequest request,
        HttpServletResponse response
    ) throws ServletException, IOException
    {
```

```
        String sEvent;
        String sRater;
        String sRatee;
        int iRateeType;
        String sRating;
```

```
        int raterID = 0;
```

```
        // get parameters
        sEvent = request.getParameter("action");
```

```
        // get stream for output
```

```

ServletOutputStream out;
response.setContentType("text/plain");
response.setHeader("Pragma", "no-cache");
response.setHeader("Cache-control", "no-cache");
response.setHeader("Expires", "0");

out = response.getOutputStream();

try
{
    DBConnection conn = new DBConnection();

    if (sEvent.equals("INIT"))
    {
        sRater = request.getParameter("rater");
        sRatee = request.getParameter("ratee");
        iRateeType = Integer.parseInt(
request.getParameter("ratee_type") );

        int rating      = -1; // not rated
        boolean implicit = false;

        String sql = "";

        // SONG case
        if (iRateeType == Constants.ITEM_TYPE_SONG)
        {
            sql = "exec sp_lcGetSongInfoSummary_xsxx "

+ sRater + ","
+ sRatee;

        }
        else if (iRateeType == Constants.ITEM_TYPE_ALBUM)
        {
            sql = "exec sp_lcGetArtistOrAlbumRating_xsxx "

+ sRatee + ","
+ sRater;

        }
        else
        {
            sql = "exec sp_lcGetArtistOrAlbumRating_xsxx "

+ sRatee + ","
+ sRater;

        }

        DBResultSet rs = conn.executeSQL(sql);

        if (!rs.getBOF() && !rs.getEOF())

```

```

        rating = rs.getInt("rating");

        out.println("rating_value=" + rating +
"&Implicit_indicator=not_implicit");
    }
    else if (sEvent.equals("RATING_EVENT"))
    {
        /* Do update to LaunchCast Ratings Database */

        sRater      = request.getParameter("rater");

        try
        {
            raterID = Integer.parseInt(sRater);
        }
        catch (NumberFormatException e)
        {
            throw new Exception("RatingWidgetServlet:
rating received for invalid user: " + sRater);
        }

        if (raterID <= 0)
        {
            throw new Exception("RatingWidgetServlet:
rating received for invalid user: " + raterID);
        }

        sRatee      = request.getParameter("ratee");
        iRateeType = Integer.parseInt(
request.getParameter("ratee_type"));
        sRating      = request.getParameter("rating");

        // song case
        if (iRateeType == Constants.ITEM_TYPE_SONG)
        {
            conn.executeUpdate("exec sp_lcRateSongUser_isux
"
+ raterID + ","
+ sRatee + ","
+ sRating, true);
        }
        // album case
        else if (iRateeType == Constants.ITEM_TYPE_ALBUM)
        {
            conn.executeUpdate("exec sp_lcRateItemUser_isux
"
+ raterID + ","
+ sRatee + ","

```

```

+ sRating, true);
    }
    // artist case
    else
    {
        conn.executeUpdate("exec sp_lcRateItemUser_isux
"
+ raterID + ","
+ sRatee + ","
+ sRating, true);
    }

    out.println("confirmation=rating_confirmed");

    if (cachedUsers.contains(new Integer(raterID)))
    {
        CachedRating cr = new CachedRating(raterID,
Integer.parseInt(sRatee), Byte.parseByte(sRating), (byte)iRateeType);
        dirtyRatings.addElement(cr);
        Util.debug("Added change to ratings cache
update queue : " + cr);
    }
    else
    {
        out.println("error");
    }

    conn.close();
}
catch(DBException e) {
    out.println("DBException: " + e.getMessage());
    System.err.println(new Date().toString() + " DBException in
RatingWidgetServlet: " + e.toString());
}
catch(Exception e) {
    out.println("Exception raised: " + e);
    System.err.println(new Date().toString() + " Exception in
RatingWidgetServlet: " + e.toString());
}

    out.close();
}

public void init (ServletConfig config)
    throws ServletException {
    super.init(config);
    try {

        gtu = new GetRatingsCacheUsers(this);
        gps = new GetPlaylistServers(this);

```



```

// the following 2 lines are for testing purposes only, rem
them out otherwise.
    //          gtu.SLEEP_TIME=1*20*1000;
    //          gps.SLEEP_TIME=1*20*1000;

    gps.start();
    gtu.start();

    myThread = new Thread(this);
    myThread.start();

}
catch (Exception e) { throw new ServletException (); }
}

/**
 * Destroy method -
 * get rid of the api
 * servlets "should have" a destroy method for garbage collection
 */

public void destroy() {

    gps.stop();
    gtu.stop();
}

//-----

public void updateCachedUsers(Vector topUsers)
{
    cachedUsers = topUsers;
}

public void updatePlaylistServers(Vector v)
{
    playlistServers = v;
}

public void run()
{
    // once every N minutes go update all cached ratings with some
new ratings

    Util.debug("RatingWidgetServlet notify playlistgenerators of
changed rating - thread started");

    try
    {
        Vector temp_dirty_ratings;
        Enumeration enum;

```

```

        Socket s;
        ByteArrayOutputStream baos;
        ObjectOutputStream oos;
        OutputStream os;
        BufferedWriter bw;
        byte b[];
        String server_ip = null;

        while (dirtyRatings != null)
        {
            try
            {
                if (dirtyRatings.size() > 0)
                {
                    baos = new ByteArrayOutputStream(1000);
                    oos = new ObjectOutputStream(baos);

                    temp_dirty_ratings = dirtyRatings;
                    dirtyRatings = new
Vector(Constants.RATING_UPDATE_LIST_INITIAL_SIZE);

                    // need to send info to cached servers
here.

                    oos.writeObject(temp_dirty_ratings);
                    oos.flush();
                    b=baos.toByteArray();

                    enum = playlistServers.elements();

                    while (enum.hasMoreElements())
                    {
                        try // this nested try / catch is
so if one server is down the others get updated too.
                        {

                            server_ip=(String)enum.nextElement();

                                Util.debug(new
Date().toString() + " RatingWidgetServlet: Sending changed ratings to : " +
server_ip + " this vector : " + temp_dirty_ratings);

                                s=new Socket(server_ip,
Constants.PORT_NUMBER);

                                os=s.getOutputStream();
                                bw=new BufferedWriter(new
OutputStreamWriter(os));

                                bw.write(Constants.POST_HEADER);

                                bw.newLine();

                                bw.write(com.launch.misc.constants.USER_AGENT + ": " +

```

```

com.launch.misc.constants.RATING_WIDGET);

b.length);

        bw.newLine();

        bw.write("Content-length: " +

        bw.newLine();
        bw.newLine();
        bw.flush();

        os.write(b);
        os.flush();
        os.close();
    }
    catch (Exception e1)
    {
        System.err.println((new
Date()).toString() + " Error contacting ratings cache at " + server_ip);
        //e1.printStackTrace();
    }
    }

    }
    catch (Exception e2)
    {
        System.err.println((new Date()).toString() + "
Error in RatingWidgetServlet CacheUpdater while loop");
        e2.printStackTrace();
    }

    Thread.sleep(Constants.PROPAGATE_DIRTY_RATING_SLEEP_TIME);
    }
    catch (Exception e)
    {
        System.err.println(new Date().toString() + " Fatal Error in
RatingWidgetServlet updater thread ");
        e.printStackTrace();
    }

    Util.debug("RatingWidgetServlet notify playlistgenerators of
changed rating - thread done");
}

    public Hashtable getMostFrequentlyUsedUsers(int i)
    {
        return null;
    }
}

```

/* eof */

D:\My Documents\email\Launch\PlaylistGenerator\RatingWidgetServlet.java Page 7 of 7 11/05/99 1:35 PM

```

package com.launch.PlaylistGenerator;

import java.util.Vector;

/**
 * Launch Media, Inc Copyright 1999
 *
 * Recommendation List - class which encapsulates
 * recommendations coming from the net perceptions engine
 *
 * RECOMMENDED USAGE
 * to access values within a RecList object:
 *
 *
 * void someFunction(RecList aRec) {
 *
 *     if ( aRec.setToFirstRec() ) {
 *         do {
 *             System.out.println( aRec.getIdentifier() + " : " +
 * aRec.getPredictedRating() );
 *             } while aRec.increment() ;
 *         }
 *     }
 *
 *
 *
 * The "prediction result" object in net perceptions is NOT
 * persistent so is unusable outside of a carefully controlled
 * environment
 *
 * Further, developers within LAUNCH should not be exposed
 * to Net Perceptions data structures (as they are ugly)
 *
 * file: launchNetP.java
 * @author Jon Heiner
 * @since 7-30-99
 */

public class RecList {

    private final static int kGrowVectorBy = 4;
    private Vector theRecs;
    private int theNumRecs = 0;
    private int theIndex = 1;

    /* Rec -- inner class
     * encapsulates the ID and predicted
     * value for the item in the list;
     * the inner values are made public
     * for convenience; they are exposed
     * to this class, but are not intended
     * to be used outside of this implementation
     */
    public class Rec {
        public int theID;

```

```

        public float theValue;

        /* Rec - creation method
         * the variables should be immutable
         */
        public Rec(int iID, float fValue) {
            theValue = fValue;
            theID = iID;
        }
    }

    /** RecList - creation method
     * creates an empty rec list, which we will then add
     * Recs to; if you try to pull values from this it will
     * indicate that this is not possible
     */
    public RecList() {
        theRecs = new Vector(0, kGrowVectorBy); // create an empty vector
    }

    /** RecList - creation method w/ args
     * creates a rec list with one element; use the add
     * method to add more values to it
     */
    public RecList(int iID, float fValue) {
        theRecs = new Vector(0, kGrowVectorBy); // create an empty vector
        this.add(iID, fValue);
    }

    /** compact
     * called once the RecList has been created and
     * all items are added
     */
    public void compact() {
        theRecs.trimToSize();
    }

    /** setToFirstRec
     * called to set us to the first rec
     * if this returns false, then there are
     * no recommendations in the list.
     */
    public boolean setToFirstRec() {
        theIndex = 0;
        if (theNumRecs > 0) return true;
        return false;
    }

    /** increment
     * moves the internal pointer to the next item
     * returns false if there are no more Recs in
     * the list.
     */
    public boolean increment() {
        theIndex++;
    }

```

```

        if (theIndex < theNumRecs) return true;
        return false;
    }

    /** getIdentifier
     *  returns the item ID for the current item
     *  in the Rec List
     */
    public int getIdentifier() {
        return (int) ((Rec) theRecs.elementAt(theIndex)).theID;
    }

    /** getPredictedRating
     *  returns the percentage value which is the
     *  predicted value
     */
    public float getPredictedRating() {
        return (float) ((Rec) theRecs.elementAt(theIndex)).theValue;
    }

    /** add
     *  adds a new value to the Rec list
     *  returns false if the values entered
     *  are invalid; (e.g.: iId < 0)
     */
    public void add(int iID, float fValue) {
        theNumRecs++;
        theRecs.addElement(new Rec(iID, fValue) );
    }

    /** length
     *  returns the number of elements in the Rec list
     */
    public int length() {
        return theNumRecs;
    }

    /** createStubRecList
     *  used to return "good" bogus values rather
     *  than values generated from Net Perceptions
     *  useful for testing and stubbing
     */
    public static RecList createStubRecList() {
        RecList aRecList = new RecList(74082, (float) 0.5);
        aRecList.add(116377, (float) 0.6);
        aRecList.add(123312, (float) 0.7);
        aRecList.add(899, (float) 0.8);
        aRecList.add(58075, (float) 0.9);

        return aRecList;
    }

    /** test
     *  test class
     */
    public static class Test {

```

```

/*
public static void main(String [] args) {
    System.out.println( "debug 0");
    RecList aRec = createStubRecList();

    System.out.println( "debug 1");
    if ( aRec.setToFirstRec() ) {
        System.out.println( "debug 2");
        do {
            System.out.println( "debug 3");
            System.out.println( aRec.getIdentifier() + " : " +
aRec.getPredictedRating() );
            System.out.println( "debug 4");
        } while ( aRec.increment() );
    }
}
*/
}
}

```

D:\My Documents\email\Launch\PlaylistGenerator\RecList.java Page 4 of 4 11/05/99 1:26 PM

```

package com.launch.PlaylistGenerator;

import java.util.Vector;
import java.util.Date;

public class SaveClips extends Thread
{
    Vector clips;
    String storedProc;
    int ordinal;
    short mediaType;
    int userID;

    public SaveClips(Vector clips, String storedProc, int ordinal, short
mediaType, int userID)
    {
        this.clips      = clips;
        this.storedProc = storedProc;
        this.mediaType  = mediaType;
        this.userID     = userID;
        this.ordinal    = ordinal;
    }

    public void run()
    {
        Date startDate = new Date();
        Thread.currentThread().setName("SaveClips for " + storedProc);

        int rowCount = 0;

        if (clips.size() <= 0)
            return;

        try
        {
            DBConnection conn = new DBConnection();
            String sql = "";

            Clip aClip;

            for (int i = 0; i < clips.size(); i++)
            {
                aClip = (Clip) clips.elementAt(i);

                sql = sql.concat(" exec " + storedProc + " "
                                + ordinal + ", "
                                + aClip.media.getID(mediaType) + ", "
                                + mediaType + ", "
                                + userID);

                ordinal++;
                rowCount++;
            }
        }
    }
}

```



```

        conn.executeSQL(sql);
        conn.close();
    }
    catch (DBException oops)
    {
        Util.debug("DB Exception: " + oops.getMessage());
    }

    Util.debug(Thread.currentThread().getName() + " saved " + rowCount + "
clips");
    Util.printElapsedTime(Thread.currentThread().getName(), startDate);
}
}

```

D:\My Documents\email\Launch\PlaylistGenerator\SaveClips.java Page 2 of 2 11/05/99 1:25 PM

```

package com.launch.PlaylistGenerator;

import java.util.Date;

public class SavePlaylist extends Thread
{
    Playlist list;
    int ordinal, to, from;

    public SavePlaylist(Playlist list, int from, int to, int ordinal)
    {
        this.list = list;
        this.ordinal = ordinal;
        this.to = to;
        this.from = from;
    }

    public void run()
    {
        Date startDate = new Date();
        Thread.currentThread().setName("SavePlaylist (" + from + "-" + to +
        ")");

        int rowCount = 0;

        try
        {
            DBConnection conn = new DBConnection();
            String sql = "";

            SongData data;
            short origin;

            for (int i = from; i < to; i++)
            {
                data = (SongData) list.media.elementAt(i);

                if (list.popularOnly)
                    origin = (short) SongData.SOURCE_FORCED_POPULAR;
                else
                    origin = (short) data.origin();

                if (data.querySource == SongData.SOURCE_RATED)
                    origin = (short) data.rating.getSource();

                //
                sql = sql.concat(" exec sp_lcSaveMediaPlaylist_ixxd "
                    + ordinal + ", "
                    + data.getMediaID(list.mediaType) + ", "
                    + list.mediaType + ", "
                    + list.userID + ", "
                    + data.implicit + ", "
                    + origin);

                ordinal++;
            }
        }
    }
}

```

```

        rowCount++;
    }
    conn.executeSQL(sql);
    conn.close();
}
catch (DBException oops)
{
    Util.debug("DB Exception: " + oops.getMessage());
}

Util.debug(Thread.currentThread().getName() + " saved " + rowCount + "
songs");
Util.printElapsedTime(Thread.currentThread().getName(), startDate);
}
}

```

D:\My Documents\email\Launch\PlaylistGenerator\SavePlaylist.java Page 2 of 2 11/05/99 1:25 PM

```

package com.launch.PlaylistGenerator;

import java.io.Serializable;

public class SimpleClip implements Serializable
{
    int mediaID;
    int ID;
    byte origin;

    public String toString()
    {
        return "clipID=" + ID + ", mediaID=" + mediaID + ", origin=" + origin;
    }

    /**
     * Constructor for ads, news, tips
     */
    public SimpleClip(int ID, int mediaID)
    {
        this.mediaID = mediaID;
        this.ID = ID;
    }

    /**
     * Constructor for songs
     */
    public SimpleClip(int ID, int mediaID, byte origin)
    {
        this(ID, mediaID);
        this.origin = origin;
    }
}

```

D:\My Documents\email\Launch\PlaylistGenerator\SimpleClip.java Page 1 of 1 11/05/99 1:32 PM

```
package com.launch.PlaylistGenerator;

import java.util.Vector;

public class SimpleClipList extends Vector
{
```

```
    public SimpleClipList(int size)
    {
        super(size);
    }
```

```
    public SimpleClip pop()
    {
        if (size() > 0)
        {
            SimpleClip clip = (SimpleClip) elementAt(0);
            removeElementAt(0);
            return clip;
        }

        return null;
    }
```

```
}
```

D:\My Documents\email\Launch\PlaylistGenerator\SimpleClipList.java Page 1 of 1 11/05/99 1:32 PM

```

package com.launch.PlaylistGenerator;

import java.util.Vector;
import java.io.Serializable;
import java.io.ByteArrayOutputStream;
import java.io.ObjectOutputStream;
import java.io.ObjectInputStream;
import java.io.ByteArrayInputStream;
import java.util.Date;

public class SimplePlaylist implements Serializable
{
    SimpleClipList news = new SimpleClipList(10);
    SimpleClipList ads = new SimpleClipList(10);
    SimpleClipList tips = new SimpleClipList(10);
    SimpleClipList songs = new SimpleClipList(50);

    Date lastAd;
    Date lastNews;
    Date lastTip;

    short mediaType;
    int moodID;
    int djID;

    public String toString()
    {
        return "ads=" + ads.toString() + ", " +
            "news=" + news.toString() + ", " +
            "songs=" + songs.toString() + ", " +
            "tips=" + tips.toString();
    }

    public void resetDates(Date newDate)
    {
        lastAd = lastNews = lastTip = newDate;
    }

    public void save(int userID)
    {
        try
        {
            DBConnection conn = new DBConnection();
            save(conn, userID);
        }
        catch (DBException e)
        {
            System.err.println(new Date().toString() + " DBException in
SimplePlaylist:save: " + e.toString());
            e.printStackTrace();
        }
    }
}

```

```

public void save(DBConnection conn, int userID)
{
    try
    {
        String sql = "exec sp_lcSavePlaylist_ixxd " + userID + ", ?";
        DBPreparedStatement statement = conn.prepareStatement(sql);
        byte[] b = toByteArray();
        statement.setBytes(1, toByteArray());
        statement.executeUpdate();
    }
    catch (DBException e)
    {
        System.err.println(new Date().toString() + " DBException in
SimplePlaylist:save:" + e.toString());
    }
}

public static SimplePlaylist fromBytes(byte[] b)
{
    if (b == null || b.length <= 0)
        return null;

    try
    {
        ByteArrayInputStream bais = new ByteArrayInputStream(b);
        if (bais == null)
            return null;

        ObjectInputStream ois = new ObjectInputStream(bais);
        if (ois == null)
            return null;

        return (SimplePlaylist) ois.readObject();
    }
    catch (Throwable e)
    {
        System.err.println("Exception in SimplePlaylist:fromBytes:" +
e.toString());
    }

    return null;
}

public static SimplePlaylist load(DBConnection conn, int userID)
{
    String sql = "exec sp_lcGetPlaylist_xsxx " + userID;

```

```

        try
        {
            DBResultSet rs = conn.executeQuery(sql);

            return SimplePlaylist.fromBytes(rs.getBytes("playlist"));
        }
        catch (Throwable e)
        {
            System.err.println("Exception in SimplePlaylist:load:" +
e.toString());
        }

        return null;
    }

    private byte[] toByteArray()
    {
        try
        {
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            ObjectOutputStream oos      = new ObjectOutputStream(baos);
            oos.writeObject(this);

            return baos.toByteArray();
        }
        catch (Throwable t)
        {
            System.err.println("toByteArray died: " + t.toString());
            t.printStackTrace();
            return null;
        }
    }
}

```

D:\My Documents\email\Launch\PlaylistGenerator\SimplePlaylist.java Page 3 of 3 11/05/99 1:35 PM


```

package com.launch.PlaylistGenerator;

public class SongData
{
    int songID;
    byte querySource;

    public AverageRating djsAverage;

    double score,
        netp,
        implicit,
        confidence,
        lastPlayed,
        bds,
        ratingF,
        djsF,
        netpF,
        commRatingF,
        lastPlayedF,
        bdsF;

    private SongInfo info;

    private Rating djs = new Rating((short) Constants.DEFAULT_DJS_SCORE);
    private byte djSource = SOURCE_DJS;

    public SongRating rating;

    PickStatus status;

    public final static byte SOURCE_RATED = 1;
    public final static byte SOURCE_IMPLICIT_ALBUM = 2;
    public final static byte SOURCE_IMPLICIT_ARTIST = 3;
    public final static byte SOURCE_IMPLICIT_SONG = 4;
    public final static byte SOURCE_DJS = 5;
    public final static byte SOURCE_DJS_SONG = 5;
    public final static byte SOURCE_BDS = 6;
    public final static byte SOURCE_POPULAR = 7;
    public final static byte SOURCE_RANDOM = 8;
    public final static byte SOURCE_NETP = 9;
    public final static byte SOURCE_ALL = 10;
    public final static byte SOURCE_RECENTLY_PLAYED = 11;
    public final static byte SOURCE_FORCED_POPULAR = 12;
    public final static byte SOURCE_GENRES = 13;
    public final static byte SOURCE_DJS_ALBUM = 14;
    public final static byte SOURCE_DJS_ARTIST = 15;

    public final static byte DO_NOTHING = 0;
    public final static byte MAKE_ME_IMPLICIT = 1;
    public final static byte EXCLUDE_ME = 2;

    public SongData(int songID)
    {

```

```

        lastPlayed = Constants.DEFAULT_LASTPLAYED_SCORE;
        djsAverage = new AverageRating((short) Constants.DEFAULT_DJS_SCORE);
        status      = new PickStatus();
        netp         = Constants.DEFAULT_NETP_SCORE;
        this.songID = songID;
        rating       = new SongRating();
    }

    public boolean equals(SongData otherData)
    {
        return (songID == otherData.songID);
    }

    public byte origin()
    {
        double maxValue = 0;
        byte maxSource   = SOURCE_RANDOM;
        byte ratingSource = 0;

        if (rating.isSet())
            ratingSource = rating.getSource();

        if (info.commRating > maxValue && info.commRating >
Constants.POPULAR_THRESHOLD && ratingSource != 1)
        {
            maxValue = info.commRating;
            maxSource = SOURCE_POPULAR;
        }

        if (djs.isSet() && djs.get() >= maxValue && djs.get() > 0 &&
ratingSource != 1)
        {
            maxValue = djs.get();
            maxSource = djSource;
        }

        /*
        if (netP > maxValue)
        {
            maxValue = netP;
            maxSource = SOURCE_NETP;
        }
        */

        if (bds > 0 && bds >= maxValue && ratingSource != 1)
        {
            maxValue = bds;
            maxSource = SOURCE_BDS;
        }

        // according to the weight matrix, if there's an explicit rating,
        //that's the only source

```

```

// but let's lie to people because they don't like it when we say
// we played lowly-rated songs for them
// even though that's what we say we will play anyway

if (rating.isSet())
{
    short value = rating.get();

    if (value > Constants.MIN_RATING_FOR_RATED_SOURCE && value >=
maxValue)
    {
        maxValue = value;
        maxSource = ratingSource;
    }
}

// lies, lies, lies.

if (maxValue < Constants.MIN_RATING_FOR_RATED_SOURCE)
{
    maxSource = SOURCE_RANDOM;
}

return maxSource;
}

public void calculateDJs(ItemsProfile items, AlbumArtistData albumAndArtist)
{
    // put in the default
    djs.set(djsAverage.get());
    djSource = SOURCE_DJS_SONG;

    if (djsAverage.count() <= 0)
    {
        djSource = SOURCE_RANDOM;

        Item albumItem = albumAndArtist.getAlbum(items, this);
        Item artistItem = albumAndArtist.getArtist(items, this);

        // don't calculate implicit ratings based on various artists
        if (artistItem != null &&
ArtistInfo.isVariousArtists(artistItem.itemID))
        {
            artistItem = null;
        }

        if (albumItem != null && albumItem.djsAverage.count() > 0)
        {
            djs.set(albumItem.djsAverage.get());
            djSource = SOURCE_DJS_ALBUM;
        }
    }
}

```

```

        else if (artistItem != null && artistItem.djsAverage.count() > 0)
        {
            djs.set(artistItem.djsAverage.get());
            djSource = SOURCE_DJS_ARTIST;
        }
    }

    public byte calculateImplicit(ItemsProfile items, AlbumArtistData
albumAndArtist)
    {

        if (!rating.isSet())
        {
            Item albumItem = albumAndArtist.getAlbum(items, this);
            Item artistItem = albumAndArtist.getArtist(items, this);

            // don't calculate implicit ratings based on various artists
            if (artistItem != null &&
ArtistInfo.isVariousArtists(artistItem.itemID))
            {
                artistItem = null;
            }

            if (albumItem != null && albumItem.userRating.isSet())
            {
                short albumRating = albumItem.userRating.get();

                if (albumRating == 0)
                    return EXCLUDE_ME;
                else
                {
                    rating.set(albumRating,
SongRating.RATING_SOURCE_FROM_ALBUM);
                    return MAKE_ME_IMPLICIT;
                }
            }
            else if (artistItem != null && artistItem.userRating.isSet())
            {
                short artistRating = artistItem.userRating.get();

                if (artistRating == 0)
                    return EXCLUDE_ME;
                else
                {
                    rating.set(artistRating,
SongRating.RATING_SOURCE_FROM_ARTIST);
                    return MAKE_ME_IMPLICIT;
                }
            }
            else if (artistItem != null && artistItem.songAverage.count() >
0)
            {

```

```

        rating.set((short)
artistItem.songAverageScore(info.album.artist),
SongRating.RATING_SOURCE_AVERAGE_SONG_RATING_BY_ARTIST);
        return MAKE_ME_IMPLICIT;
    }
}

return DO_NOTHING;
}

public void setBDS(short score)
{
    bds = score;
}

public double getBDS()
{
    return bds;
}

public void score(WeightMatrix w, StationList stations)
{
    // score bds
    bds = info.bdsScore(stations);

    byte s = rating.getSource();

    /*
    // we're not using confidence right now. Take it out for speed

    confidence = 0;

    if (ratingSource != SongRating.RATING_SOURCE_EXPLICIT)
    {
        if (djs != DEFAULT_DJS_SCORE)
            confidence += 10;
        if (netp > 0)
            confidence += 10;
        if (info.commRating > 0)
            confidence += 10;
    }
    */

    // implicit rating is based on ratings data
    ratingF      = (rating.get()      * w.matrix[s][WeightMatrix.RATING
));
    djsF         = (djs.get()         * w.matrix[s][WeightMatrix.DJS
));
    netpF        = (netp              * w.matrix[s][WeightMatrix.NETP

```

```

    });
        commRatingF = (info.commRating *
w.matrix[s][WeightMatrix.COMM_RATING]);
        lastPlayedF = (lastPlayed *
w.matrix[s][WeightMatrix.LAST_PLAYED]);
        bdsF = (bds * w.matrix[s][WeightMatrix.BDS
    });

        implicit = ratingF + djsF + netpF + commRatingF;

        // score is based on other factors
        score = implicit + lastPlayedF + bdsF;
//        confidence += w.matrix[s][WeightMatrix.CONFIDENCE];
    }

    public void setInfo(SongInfo stuff)
    {
        info = stuff;
    }

    public SongInfo getInfo()
    {
        return info;
    }

    public boolean isInfoSet()
    {
        return (info != null);
    }

    public int getArtistID()
    {
        return info.album.artist.ID;
    }

    public int getAlbumID()
    {
        return info.album.ID;
    }

    public String getArtistName()
    {
        return info.album.artist.title;
    }

    public String getAlbumName()
    {
        return info.album.title;
    }

    public int getMediaID(short mediaType)
    {
        return info.media.getID(mediaType);
    }

```

```

    }

    public String getSongName()
    {
        return info.title;
    }

    public String sourceString(byte source)
    {
        switch (source) {
            case SOURCE_RECENTLY_PLAYED:
                return "recent";
            case SOURCE_RATED:
                return "rated";
            case SOURCE_IMPLICIT_ALBUM:
                return "album";
            case SOURCE_IMPLICIT_ARTIST:
                return "artist";
            case SOURCE_IMPLICIT_SONG:
                return "s avg";
            case SOURCE_DJS:
                return "djs";
            case SOURCE_DJS_ALBUM:
                return "djAlb";
            case SOURCE_DJS_ARTIST:
                return "djArt";
            case SOURCE_BDS:
                return "bds";
            case SOURCE_POPULAR:
                return "pop";
            case SOURCE_RANDOM:
                return "random";
            case SOURCE_NETP:
                return "netp";
            case SOURCE_GENRES:
                return "genres";
            case SOURCE_ALL:
                return "all";
            default:
                return "?";
        }
    }

    public static String originText(byte origin, String singularDJ, String
    possessiveDJ)
    {
        switch (origin)
        {
            case SOURCE_RATED:
                return (singularDJ + " rated this song");
            case SOURCE_IMPLICIT_ALBUM:
                return (singularDJ + " rated this album");
            case SOURCE_IMPLICIT_ARTIST:
                return (singularDJ + " rated this artist");
            case SOURCE_IMPLICIT_SONG:

```

```

        return (singularDJ + " rated other songs by this artist");
    case SOURCE_DJS:
        return (possessiveDJ + " DJs rated this song");
    case SOURCE_DJS_ALBUM:
        return (possessiveDJ + " DJs rated this album");
    case SOURCE_DJS_ARTIST:
        return (possessiveDJ + " DJs rated this artist");
    case SOURCE_BDS:
        return (possessiveDJ + " radio stations play this song");
    case SOURCE_POPULAR:
        return "This song is popular on LAUNCHcast stations";
    case SOURCE_RANDOM:
        return "This song is a random pick";
    case SOURCE_NETP:
        return "Song recommendations";
    case SOURCE_FORCED_POPULAR:
        return "Popular - choose more genres for your music.";
    }

    return "";
}

public String toString()
{
    return "songID:" + songID + ", "
        + "score:" + score + ", "
        + "implicit:" + implicit + ", "
        + "confidence: " + confidence + ", "
        + "lastPlayed:" + lastPlayed + ", "
        + "rating:" + rating + ", "
        + "ratingSource:" + rating.getSource() + ", "
        + "bds:" + bds + ", "
        + "djs:" + djs.get() + ", "
        + "source:" + sourceString(querySource) + Util.newLine;
}

public PlaylistEntry toPlaylistEntry(short mediaType)
{
    PlaylistEntry result = new PlaylistEntry();

    result.albumID      = getAlbumID();
    result.artistID     = getArtistID();
    result.albumTitle   = info.album.title;
    result.artistTitle  = info.album.artist.title;
    result.filepath     = info.media.getFilepath(mediaType);
    result.mediaID      = getMediaID(mediaType);
    result.songID       = songID;
    result.songTitle    = info.title;
    result.title        = info.title;

    return result;
}

```



```

    }

    public SimpleClip toSimpleClip(short mediaType)
    {
        return new SimpleClip(songID, getMediaID(mediaType), origin());
    }

    public String toDisplayString(int displayType, int count)
    {
        String delim    = "";
        String prefix    = "";
        String suffix    = "";
        String bgcolor   = "";

        if (displayType == Util.DISPLAY_HTML)
        {
            if (count % 2 == 0)
                bgcolor = "#CCCCFF";
            else
                bgcolor = "white";

            delim = "</FONT></TD><TD BGCOLOR=" + bgcolor + "><FONT SIZE=\"-2\">";

            prefix = "<TR><TD BGCOLOR=" + bgcolor + "><FONT SIZE=\"-2\">";
            suffix = "</FONT></TD></TR>";
        }
        else {
            delim = "\t";
        }

        return (prefix + count
            + delim + songID
            + delim + sourceString(querySource)
            + delim + sourceString(origin())
            + delim + status.toDisplayString(displayType)
            + delim + status.order
            + delim + Util.fix(score, 2, 0)
            + delim + Math.round(lastPlayed) + "/" + Math.round(lastPlayedF)
            + delim + Math.round(bds) + "/" + Math.round(bdsF)
            + delim + Math.round(implicit)
            + delim + Util.fix(rating.get(), 0, 2) + "/" + Util.fix(ratingF,
0, 2) + " (" + rating.getSource() + ") "
            + delim + Math.round(djs.get()) + "/" + Math.round(djsF)
            + delim + Math.round(netp) + "/" + Math.round(netpF)
            + delim + Math.round(info.commRating) + "/" +
Math.round(commRatingF)
            + delim + getAlbumID()
            + delim + getArtistID()
            + delim + getArtistName()
            + delim + getSongName()
            + delim + getAlbumName()
            + delim + info.album.genresString()

```

```

        + suffix
    );
}

public String originTclList()
{
    return "{" + songID + " " + origin() + " " + Math.round(implicit) + "}"
";
}

public static String[] namesArray()
{
    String[] names = { "#",
                        "songID",
                        "query",
                        "origin",
                        "status",
                        "ord",
                        "score",
                        "lastP.",
                        "bds",
                        "impl.",
                        "rating(t)",
                        "djs",
                        "netP.",
                        "comm",
                        "albumID",
                        "artisID",
                        "artist",
                        "title",
                        "album",
    };

    return names;
}
}

```

D:\My Documents\email\Launch\PlaylistGenerator\SongData.java

Page 10 of 10 11/05/99 1:24 PM

```

package com.launch.PlaylistGenerator;

import java.util.Vector;

public class SongGroup extends Vector
{
    public SongData pickRandom(int factor)
    {
        int leftInList = size();

        if (leftInList <= 0)
            return null;

        double rand          = Util.random(leftInList - 1) + 0.00001;
        int pickIndex        = (int) Math.round((Math.pow(rand, factor) /
Math.pow(leftInList - 1, factor)) * (leftInList - 1));
        SongData pick        = (SongData) elementAt(pickIndex);
        double pickDouble    = pickIndex;
        pick.status.percentile = (short) Math.round((pickDouble / size()) *
100);

        removeElementAt(pickIndex);

        return pick;
    }
}

```

D:\My Documents\email\Launch\PlaylistGenerator\SongGroup.java Page 1 of 1 11/05/99 1:28 PM

```

package com.launch.PlaylistGenerator;

import java.util.Vector;

public class SongInfo
{
    int songID;
    byte commRating = Constants.DEFAULT_COMMRATING;
    private boolean explicit = false;

    AlbumInfo album;
    String title;
    private Vector bdsRanks;
    public MediaList media;

    public SongInfo(int songID)
    {
        this.songID = songID;
        media = new MediaList();
    }

    public void addBDSRank(BDSRank rank)
    {
        if (bdsRanks == null)
            bdsRanks = new Vector(1, 1);

        bdsRanks.addElement(rank);
    }

    public int getArtistID() /* throws Exception */
    {
        return album.artist.ID;

        /*
        if (album == null)
        {
            throw new Exception("album is not set for SongInfo songID " +
songID + "(" + title + ")");
        }

        return album.getArtistID();
        */
    }

    public int getAlbumID() /* throws Exception */
    {
        /*
        if (album == null)
        {
            throw new Exception("album is not set for SongInfo songID " +
songID + "(" + title + ")");
        }
        */
    }

```

```

        return album.ID;
    }

    public double bdsScore(StationList stations)
    {
        if (bdsRanks == null || stations.size() <= 0)
            return Constants.DEFAULT_BDS_SCORE;

        int i                = 0;
        int pointBar          = Constants.BDS_SCORE_POINTBAR;
        float maxPoints       = Constants.BDS_SCORE_MAX_POINTS;
        float totalpoints     = 0;
        float numStations     = 0;

        BDSRank rank;
        Station sta;

        for (int j = 0; j < bdsRanks.size(); j++)
        {
            rank = (BDSRank) bdsRanks.elementAt(j);
            sta  = stations.get(rank.stationID);

            if (sta != null)
            {
                totalpoints += (maxPoints - rank.rank);
                numStations++;
            }
        }

        double potentialStations = stations.size();

        double score = (((totalpoints / potentialStations) / maxPoints) +
            (numStations / potentialStations) ) * 150.0);

        return score;
    }

    public String bdsString()
    {
        String result = "";

        if (bdsRanks == null)
            return "(none)";

        for (int i = 0; i < bdsRanks.size(); i++)
        {
            result = result.concat(bdsRanks.elementAt(i).toString() + ",");
        }

        return "(" + result + ")";
    }

    public String toString()
    {

```

```

        return "songID=" + songID + ", "
            + "title=" + title + ", "
            + "commRating=" + commRating + ", "
            + "media=" + media.toString()
            + "bdsRanks=" + bdsString()
            + "album=" + album.toString();
    }

    public void setExplicitLyrics(boolean badStuff)
    {
        explicit = badStuff;
    }

    public boolean hasExplicitLyrics()
    {
        return explicit;
    }
}

```

D:\My Documents\email\Launch\PlaylistGenerator\SongInfo.java Page 3 of 3 11/05/99 1:35 PM

```

package com.launch.PlaylistGenerator;

import java.util.Hashtable;
import java.util.Enumeration;
import javax.servlet.ServletOutputStream;
import java.util.Date;
import java.util.Vector;

public class SongInfoCache
{
    private Hashtable songs;
    private Hashtable albums;
    private Hashtable artists;
    private SongInfo songList[];
    private Hashtable ads;
    private Hashtable news;
    private Hashtable tips;
    private Clip adList[];
    private Clip newsList[];
    private Clip tipList[];
    private IntHash mediaTypes;
    public PopularSongs popular;
    public RatingsCache ratingsCache;
    private GenreIndex genres;

    public final static byte TYPE_SONG    = 1;
    public final static byte TYPE_ALBUM   = 2;
    public final static byte TYPE_ARTIST  = 3;
    public final static byte TYPE_AD      = 4;
    public final static byte TYPE_NEWS    = 5;
    public final static byte TYPE_TIP     = 6;

    private ServletOutputStream out;

    public Date lastUpdate;

    public SongInfoCache(ServletOutputStream out)
    {
        // use memory most efficiently with load factor 1
        songs      = new Hashtable(50000);
        albums     = new Hashtable(3000);
        artists    = new Hashtable(1500);
        ads        = new Hashtable();
        news       = new Hashtable();
        tips       = new Hashtable();
        mediaTypes = new IntHash();
        genres     = new GenreIndex(100, 1);

        populate();

        lastUpdate = new Date();
    }

    public SongList getPopular(short mediaType)

```

```

    {
        return popular.get(mediaType);
    }

    public SongList getInGenres(GenreList myGenres)
    {
        return genres.getInGenreList(myGenres);
    }

    public SongList getInGenre(int genreID)
    {
        return genres.getInGenre(genreID);
    }

    public int countInGenres(GenreList myGenres)
    {
        return genres.countInGenreList(myGenres);
    }

    private void populate()
    {
        try
        {
            DBConnection conn = new DBConnection();
            DBResultSet rs = conn.executeQuery("exec
sp_lcoGetSongDataCache_xxxx");

            int songID, mediaType, rank, stationID, rowCount;
            short genreID;
            String filePath;
            SongInfo aSong;
            ArtistInfo anArtist;
            AlbumInfo anAlbum;

            rowCount = 0;

            while (!rs.getBOF() && !rs.getEOF())
            {
                songID = rs.getInt("songID");
                mediaType = rs.getInt("mediaType");

                aSong = (SongInfo) init(songID, SongInfoCache.TYPE_SONG);

                filePath = rs.getString("server") +
rs.getString("directory") + "\\\" + rs.getString("filePath");

                aSong.media.add((short) mediaType, rs.getInt("mediaID"),
filePath);

                aSong.title = rs.getString("song");

                anArtist = (ArtistInfo) init(rs.getInt("artistID"),
SongInfoCache.TYPE_ARTIST);
                anArtist.title = rs.getString("artist");
            }
        }
    }

```



```

        anArtist.songs.put(new Integer(songID), aSong);

        anAlbum = (AlbumInfo) init(rs.getInt("albumID"),
SongInfoCache.TYPE_ALBUM);
        anAlbum.title = rs.getString("album");

        aSong.setExplicitLyrics(rs.getInt("explicit") == 1);

        // add year and date added

        anAlbum.artist = anArtist;
        aSong.album = anAlbum;

        mediaTypes.increment(mediaType);

        rowCount++;
        rs.next();
    }

    Util.debug("SongInfoCache:populate loaded " + rowCount + "
media");

    rs = conn.executeQuery("exec sp_lcoGetCommRatingCache_xxxx");
    rowCount = 0;

    while (!rs.getBOF() && !rs.getEOF())
    {

        songID = rs.getInt("songID");
        aSong = (SongInfo) get(songID, SongInfoCache.TYPE_SONG);

        if (aSong != null)
        {
            aSong.commRating = (byte) rs.getInt("commRating");
            rowCount++;
        }
        rs.next();
    }

    Util.debug("SongInfoCache:populate loaded " + rowCount + "
commRatings");

    rs = conn.executeQuery("exec sp_lcoGetGenreCache_xxxx");

    while (!rs.getBOF() && !rs.getEOF())
    {

        genreID = (short) rs.getInt("genreID");
        songID = rs.getInt("songID");
        aSong = (SongInfo) get(songID, SongInfoCache.TYPE_SONG);

        if (aSong != null && aSong.album != null)
        {
            aSong.album.addGenre(genreID);
            genres.add(genreID, aSong);
        }
    }

```

```

        rowCount++;
    }
    rs.next();
}

Util.debug("SongInfoCache:populate loaded " + rowCount + " genre
mappings");
rowCount = 0;

rs = conn.executeQuery("exec sp_lcoGetBDSCache_xsxx");
while (!rs.getBOF() && !rs.getEOF())
{
    songID = rs.getInt("songID");

    aSong = (SongInfo) get(songID, TYPE_SONG);

    if (aSong != null)
    {
        rank      = rs.getInt("rank");
        stationID = rs.getInt("stationID");

        rowCount++;
        aSong.addBDSRank(new BDSRank((short) stationID,
(byte) rank));
    }

    rs.next();
}

Util.debug("SongInfoCache:populate loaded " + rowCount + " bds
Ranks");

// import ads
rowCount = 0;
rs = conn.executeQuery("exec sp_lcoGetAdCache_xsxx");

Clip ad;
int clipID;

while (!rs.getBOF() && !rs.getEOF())
{
    clipID = rs.getInt("clipID");

    //          filePath = rs.getString("server") +
rs.getString("directory") + "/" + rs.getString("filePath");

    ad      = (Clip) init(clipID, TYPE_AD);

    //          ad.name      = rs.getString("clipName");
    ad.media.add((short) rs.getInt("mediaType"),
rs.getInt("mediaID"), null);

    rowCount++;
    rs.next();
}

```

```

    }

    Util.debug("SongInfoCache:populate loaded " + rowCount + " ad
media");

    // import news

    rs = conn.executeQuery("exec sp_lcoGetNewsCache_xsxx");
    rowCount = 0;
    Clip newsbit;

    while (!rs.getBOF() && !rs.getEOF())
    {
        clipID = rs.getInt("clipID");

        filePath = rs.getString("server") +
rs.getString("directory") + "\\\" + rs.getString("filePath");

        newsbit = (Clip) init(clipID, TYPE_NEWS);

        newsbit.name = rs.getString("clipName");
        newsbit.media.add((short) rs.getInt("mediaType"),
rs.getInt("mediaID"), filePath);
        rowCount++;
        rs.next();
    }

    Util.debug("SongInfoCache:populate loaded " + rowCount + " news
media");

    // import tips

    rs = conn.executeQuery("exec sp_lcoGetTipCache_xsxx");
    rowCount = 0;
    Clip tip;

    while (!rs.getBOF() && !rs.getEOF())
    {
        clipID = rs.getInt("clipID");

        filePath = rs.getString("server") +
rs.getString("directory") + "\\\" + rs.getString("filePath");

        tip = (Clip) init(clipID, TYPE_TIP);

        tip.name = rs.getString("clipName");
        tip.media.add((short) rs.getInt("mediaType"),
rs.getInt("mediaID"), filePath);
        rowCount++;
        rs.next();
    }

    Util.debug("SongInfoCache:populate loaded " + rowCount + " tip
media");

    conn.close();

```

```

    }
    catch (DBException oops)
    {
        System.out.println("DBException in cache populate: " +
oops.getMessage());
    }

    // populate the songs array
    songList = new SongInfo[songs.size()];
    int i = 0;
    for (Enumeration e = songs.keys(); e.hasMoreElements() ;) {
        songList[i] = (SongInfo) songs.get((Integer) e.nextElement());
        i++;
    }

    // populate the ads array
    adList = new Clip[ads.size()];
    i = 0;
    for (Enumeration e = ads.keys(); e.hasMoreElements() ;) {
        adList[i] = (Clip) ads.get((Integer) e.nextElement());
        i++;
    }

    // populate the news array
    newList = new Clip[news.size()];
    i = 0;
    for (Enumeration e = news.keys(); e.hasMoreElements() ;) {
        newList[i] = (Clip) news.get((Integer) e.nextElement());
        i++;
    }

    // populate the tips array
    tipList = new Clip[tips.size()];
    i = 0;
    for (Enumeration e = tips.keys(); e.hasMoreElements() ;) {
        tipList[i] = (Clip) tips.get((Integer) e.nextElement());
        i++;
    }

    // make popular lists
    popular = new PopularSongs(songs, mediaTypes);
    Util.debug("SongInfoCache:populate done");
}

```

```

private Hashtable getHash(byte type)
{

```

```

        if (type == TYPE_SONG)
            return songs;
        else if (type == TYPE_ALBUM)
            return albums;
        else if (type == TYPE_ARTIST)
            return artists;
        else if (type == TYPE_AD)
            return ads;
        else if (type == TYPE_NEWS)
            return news;
        else if (type == TYPE_TIP)
            return tips;

        return null;
    }

    public Object init(int ID, byte type)
    {
        if (getHash(type).containsKey(new Integer(ID)))
        {
            return get(ID, type);
        }
        else {
            return put(ID, type);
        }
    }

    public Object get(Integer ID, byte type)
    {
        return (getHash(type)).get(ID);
    }

    public Object get(int ID, byte type)
    {
        return get(new Integer(ID), type);
    }

    private Object makeNew(int ID, byte type)
    {
        if (type == TYPE_SONG)
            return new SongInfo(ID);
        else if (type == TYPE_ALBUM)
            return new AlbumInfo(ID);
        else if (type == TYPE_ARTIST)
            return new ArtistInfo(ID);
        else if (type == TYPE_AD)
            return new Clip(ID, Clip.TYPE_AD);
        else if (type == TYPE_NEWS)
            return new Clip(ID, Clip.TYPE_NEWS);
        else if (type == TYPE_TIP)
            return new Clip(ID, Clip.TYPE_TIP);

        return null;
    }

    private Object put(int ID, byte type)

```

```

    {
        Hashtable hash = getHash(type);

        Object thing = makeNew(ID, type);
        hash.put(new Integer(ID), thing);
        return thing;
    }

    public SongInfo randomSong()
    {
        long index = Util.random(songList.length - 1);

        if (index > songList.length - 1)
            return null;

        return songList[(int) index];
    }

    public Enumeration keys(byte type)
    {
        if (type == TYPE_SONG)
            return songs.keys();
        else if (type == TYPE_ALBUM)
            return albums.keys();
        else if (type == TYPE_ARTIST)
            return artists.keys();
        else if (type == TYPE_AD)
            return ads.keys();
        else if (type == TYPE_NEWS)
            return news.keys();
        else if (type == TYPE_TIP)
            return tips.keys();

        return null;
    }

    public int size(byte type)
    {
        Hashtable hash = getHash(type);

        if (hash != null)
            return hash.size();

        return 0;
    }

    private Clip[] getClipList(byte type)
    {
        if (type == TYPE_AD)
            return adList;
        else if (type == TYPE_NEWS)
            return newsList;
        else if (type == TYPE_TIP)
            return tipList;
    }

```

```

        return null;
    }

    public Clip randomClip(byte type)
    {
        Clip[] clips = getClipList(type);

        if (clips == null || clips.length <= 0)
            return null;

        return clips[(int) Util.random(clips.length - 1)];
    }

    public Vector randomClipList(byte type, short mediaType, int max)
    {
        Vector list = new Vector();

        Clip bip;

        // stop if we have enough or we've iterated too many times
        for (int i = 0; i < (max * 10) && list.size() < max; i++)
        {
            int iterations = max;
            boolean cool = false;
            boolean done = false;

            do
            {
                bip = randomClip(type);
                iterations--;

                // maybe we didn't get one
                if (bip == null)
                {
                    done = true;
                }
                else
                {
                    // we got one that fits!
                    cool = (bip.media.inType(mediaType) &&
!list.contains(bip));

                    // we've got to stop sometime
                    done = (cool || iterations < 0);
                }
            }
            while (!done);

            // if it was cool, go ahead
            if (cool)
                list.addElement(bip);
        }

        return list;
    }
}

```

```

package com.launch.PlaylistGenerator;

import javax.servlet.http.HttpServlet;
import java.util.Date;

public class SongInfoCacheUpdater extends Thread
{
    PlaylistGeneratorServlet servlet;

    public SongInfoCacheUpdater(PlaylistGeneratorServlet servlet)
    {
        this.servlet = servlet;
    }

    public void run()
    {
        Thread.currentThread().setName("SongInfoCacheUpdater");

        // update every day
        long timeToSleep = Util.MILLISECONDS_IN_SECOND *
                               Util.SECONDS_IN_MINUTE *
                               Util.MINUTES_IN_HOUR *
                               Util.HOURS_IN_DAY;

        while (true)
        {
            try { Thread.sleep(timeToSleep); } catch (InterruptedException e)
            {}

            try
            {
                Util.debug("updating song cache at " + new Date());
                Util.debug("last update was at " +
servlet.songCache.lastUpdate);

                // make a new cache
                SongInfoCache cache = new SongInfoCache(null);

                // make sure to copy over the ratingsCache too!!!
                cache.ratingsCache = servlet.songCache.ratingsCache;

                // install the new cache
                servlet.songCache = cache;
                Util.debug("finished updating song cache at " + new
Date());

                Util.debug("last update is now at " +
servlet.songCache.lastUpdate);
            }
            catch (Throwable e)
            {
                System.err.println("SongInfoCacheUpdater caught an
exception: " + e.toString());
                e.printStackTrace();
            }
        }
    }
}

```



```
}  
}  
}  
}  
}
```

D:\My Documents\email\Launch\PlaylistGenerator\SongInfoCacheUpdater.java Page 2 of 2 11/05/99 1:38 PM

```

package com.launch.PlaylistGenerator;

import java.util.Vector;
import java.util.Hashtable;
import java.util.Enumeration;

public class SongList implements Cloneable
{
    private Vector    list = new Vector();
    private Hashtable unique = new Hashtable();
    private boolean ordered = false;

    public SongList()
    {
    }

    /**
     * Creates a SongList from a Hashtable of songs
     */
    public SongList(Hashtable songs)
    {
        SongInfo info = null;
        Integer songID;

        for (Enumeration e = songs.keys(); e.hasMoreElements();)
        {
            songID = (Integer) e.nextElement();
            info = (SongInfo) songs.get(songID);
            addElement(info);
        }
    }

    public SongList(Hashtable songs, short mediaType)
    {
        Integer songID;
        SongInfo info = null;

        for (Enumeration e = songs.keys(); e.hasMoreElements();)
        {
            songID = (Integer) e.nextElement();
            info = (SongInfo) songs.get(songID);

            if (info.media.inType(mediaType))
            {
                addElement(info);
            }
        }
    }

    public void addElement(SongInfo info)
    {
        Integer ID = new Integer(info.songID);
    }
}

```

```

        // check unique constraint
        if (unique.get(ID) == null)
        {
            list.addElement(info);
            unique.put(ID, info);
        }
    }

    public void addElements(SongList list)
    {
        if (list == null)
            return;

        for (int i = 0; i < list.size(); i++)
        {
            addElement(list.elementAt(i));
        }
    }

    public void sort()
    {
        sort(this, 0, list.size() - 1);

        ordered = true;
    }

    public int size()
    {
        return list.size();
    }

    public SongInfo elementAt(int index)
    {
        return (SongInfo) list.elementAt(index);
    }

    public void setSize(int newSize)
    {
        list.setSize(newSize);
    }

    private void sort(SongList a, int from, int to)
    {
        // quicksort
        // If there is nothing to sort, return
        if ((a == null) || (a.size() < 2)) return;

        int i = from, j = to;
        SongInfo center = a.elementAt((from + to) / 2);

        do {
            while((i < to) && (center.commRating <
a.elementAt(i).commRating)) i++;

```

```

        while((j > from) && (center.commRating >
a.elementAt(j).commRating)) j--;

        if (i < j) {
            SongInfo temp = a.elementAt(i);
            a.setElementAt(a.elementAt(j), i);
            a.setElementAt(temp, j);        // swap elements
        }
        if (i <= j) { i++; j--; }
    } while(i <= j);
    if (from < j) sort(a, from, j); // recursively sort the rest
    if (i < to) sort(a, i, to);

}

public void setElementAt(SongInfo info, int index)
{
    list.setElementAt(info, index);
}

public SongInfo pickRandom()
{
    if (size() <= 0)
        return null;

    int lucky = (int) Util.random(size() - 1);

    if (lucky < 0)
        return null;

    SongInfo info = elementAt(lucky);
    list.removeElementAt(lucky);

    return info;
}

public Object clone()
{
    SongList result = new SongList();
    result.ordered = this.ordered;

    result.unique = (Hashtable) unique.clone();
    result.list = (Vector) list.clone();

    return result;
}
}

```

```

package com.launch.PlaylistGenerator;

public class SongRating
{
    public final static byte RATING_SOURCE_NONE          = 0;
    public final static byte RATING_SOURCE_EXPLICIT      = 1;
    public final static byte RATING_SOURCE_FROM_ALBUM    = 2;
    public final static byte RATING_SOURCE_FROM_ARTIST   = 3;
    public final static byte RATING_SOURCE_AVERAGE_SONG_RATING_BY_ARTIST = 4;

    private short rating = (short) Constants.DEFAULT_RATING;
    private boolean set = false;
    private byte type;

    public boolean isSet()
    {
        return set;
    }

    public short set(short newRating, byte newType)
    {
        rating = newRating;
        type = newType;
        set = true;

        return rating;
    }

    public short get()
    {
        return rating;
    }

    public byte getSource()
    {
        return type;
    }
}

```

D:\My Documents\email\Launch\PlaylistGenerator\SongRating.java Page 1 of 1 11/05/99 1:38 PM

```

package com.launch.PlaylistGenerator;

public class Song
{
    public final static short EXCLUDED = 4;
    public final static short EXPLICIT = 3;
    public final static short IMPLICIT = 2;
    public final static short UNRATED = 1;
    public final static short ANY = 0;

    public int songID;
    public short type = ANY;
    private SongData data = null;

    public Song(int songID, short type)
    {
        this.songID = songID;
        setType(type);
    }

    public String toString()
    {
        return "Song " + songID
            + ", type = "
            + typeString()
            + ", data = "
            + ((data == null) ? "null" : data.toString());
    }

    public String typeString()
    {
        switch (type)
        {
            case ANY:
                return "ANY";
            case EXPLICIT:
                return "EXPLICIT";
            case IMPLICIT:
                return "IMPLICIT";
            case UNRATED:
                return "UNRATED";
            case EXCLUDED:
                return "EXCLUDED";
            default:
                return "UNKNOWN";
        }
    }

    // this should wait for setType
    public SongData getData()
    {
        return data;
    }

    // this should wait for setType

```

```

public short getType()
{
    return type;
}

// returns whether or not this is suitable for setting SongData
public boolean setType(short newType)
{
    short oldType = type;

    if (newType == type)
        return true;
    else if (newType < type)
        return false;
    else
        type = newType;

    // add or delete song data

    if (newType == EXCLUDED)
    {
        //          if (oldType != 0)
        //          Util.debug(Thread.currentThread().getName() + ": deleting
data for song " + songID + ", oldType was " + oldType);
        data = null;
    }
    else if (oldType == ANY && (newType == EXPLICIT || newType == IMPLICIT
|| newType == UNRATED))
    {
        data = new SongData(songID);
    }

    return true;
}
}

```

D:\My Documents\email\Launch\PlaylistGenerator\Song.java Page 2 of 2 11/05/99 1:26 PM

```
package com.launch.PlaylistGenerator;
```

```
public class Station
```

```
{
```

```
    int ID;
```

```
    public Station(int stationID)
```

```
    {
```

```
        ID = stationID;
```

```
    }
```

```
}
```

```
D:\My Documents\email\Launch\PlaylistGenerator\Station.java Page 1 of 1    11/05/99 1:26 PM
```



```

package com.launch.PlaylistGenerator;

import java.util.Vector;

public class StationList
{
    private Vector slist;

    public StationList()
    {
        slist = new Vector();
    }

    public Station stationAt(int i)
    {
        return (Station) slist.elementAt(i);
    }

    public void addElement(Station s)
    {
        slist.addElement(s);
    }

    public int size()
    {
        return slist.size();
    }

    public String inList()
    {
        Integer list[] = new Integer[size()];
        int last = 0;
        for (int i = 0; i < slist.size(); i++)
        {
            list[i] = new Integer(stationAt(i).ID);
        }
        return Util.join(", ", list);
    }

    public Station get(int stationID)
    {
        for (int i = 0; i < slist.size(); i++)
        {
            if (stationAt(i).ID == stationID)
            {
                return stationAt(i);
            }
        }
        return null;
    }
}

```

```

package com.launch.PlaylistGenerator;

import java.io.OutputStream;
import java.util.Date;
import javax.servlet.ServletOutputStream;
import java.io.IOException;

public class Util
{
    public static final int MILLISECONDS_IN_SECOND = 1000;
    public static final int SECONDS_IN_MINUTE = 60;
    public static final int MINUTES_IN_HOUR = 60;
    public static final int HOURS_IN_DAY = 24;
    public static final int DAYS_IN_WEEK = 7;
    public static final int DAYS_IN_MONTH = 30;

    public static final int DISPLAY_TEXT = 0;
    public static final int DISPLAY_HTML = 1;

    public static final String newLine = "\r\n";

    public static final short average(double count, double sum)
    {
        if (count == 0)
            return 0;

        return (short) Math.round(sum / count);
    }

    public static final long random(int ceiling)
    {
        return Math.round(Math.random() * ceiling);
    }

    public static final String join (String delim, Object values[])
    {
        String result = "";
        int i = 0;

        for (; i < values.length; i++)
            result = result.concat(values[i].toString() + delim);

        if (i > 0)
            result = result.substring(0, (result.length() - delim.length()));

        return result;
    }

    public static final String fix(double number, int precision, int zeroFill)
    {
        double power = Math.pow(10, precision);
        double fixed = Math.round(number * power) / power;
        String mantissa = new Long(Math.round(fixed)).toString();
    }

```

```

        String result = mantissa;

        for (int i = mantissa.length(); i < zeroFill; i++)
            result = new String("0" + result);

        return result;
    }

    public static final void out(ServletOutputStream stream, String whatever)
    {
        try
        {
            if (stream == null)
                System.out.println(whatever);
            else
                stream.println(whatever);
        }
        catch (IOException e)
        {
        }
    }

    public static final void debug(String info)
    {
        System.out.println(info);
    }

    public final static String tab(int times)
    {
        String result = "";

        for (int i = 0; i < times; i++)
        {
            result = result.concat("    ");
        }

        return result;
    }

    public static final void markQueryFinished(String threadName, Date startDate)
    {
        Util.debug(newLine + threadName + " started getting data after "
            + ((new Date().getTime() - startDate.getTime()) /
1000.0)
            + " seconds" + newLine);
    }

    public static final void printElapsedTime(String threadName, Date startDate)
    {
        Util.debug(newLine + new Date().toString() + " " + threadName + " took

```

```
"
                                + ((new Date().getTime() - startDate.getTime()) /
1000.0)
                                + " seconds" + newLine);
    }

    public static final String tab()
    {
        return tab(1);
    }
}
```

D:\My Documents\email\Launch\PlaylistGenerator\Util.java Page 3 of 3 11/05/99 1:37 PM

```
package com.launch.PlaylistGenerator;
```

```
public class WeightMatrix
```

```
{  
    public final static byte RATING      = 0;  
    public final static byte DJS        = 1;  
    public final static byte NETP       = 2;  
    public final static byte COMM_RATING = 3;  
    public final static byte LAST_PLAYED = 4;  
    public final static byte BDS        = 5;  
    public final static byte CONFIDENCE  = 6;  
  
    // rating, djs, netp, commRating, lastPlayed, bds, conf  
  
    public double matrix[][] = {  
no rating      {0.00, 0.33, 0.00, 0.10, 0.25, 0.20, 0.0}, //  
explicit rating {0.70, 0.00, 0.00, 0.00, 0.30, 0.00, 100.0}, //  
album rating only {0.45, 0.05, 0.00, 0.05, 0.20, 0.20, 50.0}, //  
artist only      {0.40, 0.10, 0.00, 0.05, 0.20, 0.20, 30.0}, //  
cross-propagated song ratings {0.35, 0.15, 0.00, 0.05, 0.20, 0.20, 20.0} //  
    };  
}
```

```
D:\My Documents\email\Launch\PlaylistGenerator\WeightMatrix.java    Page 1 of 1    11/05/99 1:32 PM
```

From: Rho, Angie
Sent: Friday, August 06, 1999 4:32 PM
To: Boulter, Jeff
Subject: LAUNCHcast Question

Hey Jeff - I think I lost all of my ratings. When I clicked on the Update to your launch.com address link, it took me to an update page. When I put in new information, then I got to a homepage with a Create your station now page to start rating again. Is all of my old data gone? Oh no . . . Angie

Subject:rating widget servlet

Date:Wed, 11 Aug 1999 10:19:02 -0800

From:Boulter, Jeff

To:Heiner, Jon

CC:Beaupre, Todd

have you changed the servlet so that it accepts ratee_types or 2 and 3 or still just 2?

Also, we should not allow ratings for these values:

```
        public final static int ARTIST_VARIOUS_ARTISTS          = 1028125;
        public final static int ALBUM_ORIGINAL_SOUNDTRACK      =
1020156;
        public final static int ALBUM_ORIGINAL_SOUNDTRACK_SCORE = 1021057;
        public final static int ALBUM_ORIGINAL_SOUNDTRACK_SELECTIONS =
1021058;
        public final static int ALBUM_ORIGINAL_SOUNDTRACK_THEMES =
1021059;
        public final static int ALBUM_ORIGINAL_SOUNDTRACK_CASTS = 1021060;
```

Jeff Boulter

Date: Thu, 2 Sep 1999 14:59:29 -0800
From: Boulter, Jeff
To: Leung, Ted

Install this:
\\ntdeptserver2\Applications\PC Apps\WNT IIS 4.0
\\jennings\download\jrun20.exe

sp_lcGetRatingsCacheServers_xsxd

```
package com.launch.PlaylistGenerator;
```

```
import java.util.Properties;  
import java.io.*;  
import javax.servlet.*;  
import javax.servlet.http.*;
```

```
/**
```

```
-----  
*  
* RatingWidgetServlet.java 7/8/99  
* Initial Servlet for ratings Widget  
* Copyright (c) 1999 Launch, Inc.  
* @author Jon Heiner  
-----  
*/
```

```
public class RatingWidgetServlet extends HttpServlet implements Constants  
{
```

```
    /**
```

```
     * Handle requests...
```

```
    */
```

```
    public void doGet (  
        HttpServletRequest request,  
        HttpServletResponse response  
    ) throws ServletException, IOException  
    {
```



```

String sEvent;
String sRater;
String sRatee;
int iRateeType;
String sRating;

// get parameters
sEvent = request.getParameter("action");

// get stream for output
ServletOutputStream out;
response.setContentType("text/plain");
out = response.getOutputStream();

try
{
    DBConnection conn = new DBConnection();

    if (sEvent.equals("INIT"))
    {
        sRater = request.getParameter("rater");
        sRatee = request.getParameter("ratee");
        iRateeType = Integer.parseInt(
request.getParameter("ratee_type") );

        int rating = -1;
        boolean implicit = false;

        String sql = "";

        // SONG case
        if (iRateeType == ITEM_TYPE_SONG)
        {
            sql = "exec
sp_lcGetSongInfoSummary_xxxx "
+ sRater + ","
+ sRatee;

            // ARTIST OR ALBUM case
        }
        else
        {
            sql = "exec
sp_lcGetArtistOrAlbumRating_xxxx "
+ sRater + ","
+ sRatee;

            DBResultSet rs = conn.executeQuery(sql);

            if (!rs.getBOF() && !rs.getEOF())
                rating = rs.getInt("rating");

            out.println("rating_value=" + rating +
"&Implicit_indicator=not_implicit");

```

```

    }
    else if (sEvent.equals("RATING_EVENT"))
    {
        /* Do update to LaunchCast Ratings

Database */

        sRater      =
request.getParameter("rater");
        sRatee      =
request.getParameter("ratee");
        iRateeType = Integer.parseInt(
request.getParameter("ratee_type") );
        sRating     =
request.getParameter("rating");

        if (iRateeType == ITEM_TYPE_SONG)
        {
            conn.executeSQL("exec
sp_lcRateSongUser_isux "
+ sRater + ","
+ sRatee + ","
+ sRating );
        }
        else
        {
            conn.executeSQL("exec
sp_lcRateItemUser_isux "
+ sRater + ","
+ sRatee + ","
+ sRating );
        }

        out.println("confirmation=rating_confirmed");
    }
    else
    {
        out.println("error");
    }

    conn.close();
}
catch (DBException e) {
    out.println("DBException: " + e.getMessage());
}
catch (Exception e) {
    out.println("Exception raised: " + e);
    e.printStackTrace();
}

out.close();
}

```

```

    /**
     * Initialization method - Initializes static engine object
     * references (performance enhancement to reuse persistent
     * CORBA objects.)
     */
    public void init (ServletConfig config)
        throws ServletException {
        super.init(config);
        try {
            //private final static String marker =
            //private final static String host =
            "prod";
            "athena";
            // api = new LaunchNetP(); // creation method does
            automatic initialization
        }
        catch (Exception e) { throw new ServletException (); }
    }

    /**
     * Destroy method -
     * get rid of the api
     * servlets "should have" a destroy method for garbage collection
     */

    public void destroy() {
        // api = null; // hopefully this causes the api to be garbage
        collected as well.
    }
}

/* eof */

```

Jeff Boulter

Subject:RE: LAUNCH Cast
Date:Tue, 7 Sep 1999 10:07:26 -0800
From:Boulter, Jeff
To:Mohler, Dan

Just remove the html part.

<http://devweb2.launch.com/music/launchcast/>

> -----Original Message-----
> From: Mohler, Dan
> Sent: Tuesday, September 07, 1999 11:06 AM
> To: LAUNCHcast Developers
> Subject: LAUNCH Cast
>
> Can I get the latest URL. This one doesn't work for me.
>
>
> <http://devweb2.launch.com/music/launchcast/home/1,3874,,FF.html>
>
> Dan Mohler
> Sr VP Advertising
> LAUNCH Media

Subject:changes

Date:Tue, 7 Sep 1999 19:45:28 -0800

From:Boulter, Jeff

To:Leung, Ted

1. SongIDs and ItemIDs are in the same number space
2. Update DJs list every 10 minutes; only add/remove user ratings if the list changed
3. Optionally dump all ratings every so often and refresh from DB

Code is in \\jennings\PlaylistGenerator

Jeff

Subject:playlist generator URL

Date:Wed, 8 Sep 1999 13:56:05 -0800

From:Boulter, Jeff

To:Leung, Ted

<http://devweb7/servlet/playlist?u=13302&debug=1&forceRefresh=1&matrix=1>

Jeff Boulter

From: Boulter, Jeff
Sent: Monday, September 13, 1999 2:59 PM
To: 'paulbou@microsoft.com'
Cc: Morrow, Greg; Beaupre, Todd
Subject: Launch and Windows Media

Hi Paul, I got the message that you called to help us with our implementation of Windows Media Technology. Here's an email that summarizes our current issues.

<<Options for personalization with Windows Media Technologies>>

If you have any questions, please feel free to give me call or send me email.

Thanks,

Jeff

From: Beaupre, Todd
Sent: Saturday, September 16, 1999 9:30 PM
To: Boulter, Jeff
Subject: FW: ItemRatings conversion algorithm --

keep a copy of this

-----Original Message-----
From: Beaupre, Todd
Sent: Saturday, September 18, 1999 8:29 PM
To: Moran, Daniel
Subject: RE: ItemRatings conversion algorithm --

agreed about the source table field.

mapping
1 - 0 hate it
2 - 15 pretty bad
3 - 30 Not my thing
4 - 45 It's OK
5 - 60 I like that
6 - 75 Great
7 - 90 The best

From: Schulte, Barbara
Sent: Tuesday, September 21, 1999 1:58 PM
To: Boulter, Jeff; Beaupre, Todd
Subject: ratings

it doesn't seem like launchcast is remembering my ratings. today i've Xd out jon secada and sean lennon, as well as some acid jazz and surf CDs, but i keep hearing songs from all of them. it's almost like i'm hearing them more often the more i X them out.

It's the same as it was the other day when it showed that something was Xd but still played it -- this time the Xs aren't even there.

thanks.

From: Lee, Howard
Sent: Wednesday, September 22, 1999 8:28 PM
To: Boulter, Jeff; Beaupre, Todd
Subject: FW: URL's

These are test URL's based on the servlets and the params we discussed last night. I dunno if this email is extraneous but hey it might be convenient who knows.

-h-

-----Original Message-----

From: Lee, Howard
Sent: Wednesday, September 22, 1999 7:51 PM
To: Lee, Howard
Subject: URL's

<http://209.67.158.17/servlet/gateway?u=6474126&debug=1> LaunchStream1
<http://209.67.158.16/servlet/gateway?u=6474126&debug=1> LaunchStream2

<http://209.67.158.18/servlet/playlist?u=6474126&d=6474126&m=0&b=0&forceRefresh=1&matrix=1&debug=1>
LCPlaylist1
<http://209.67.158.29/servlet/playlist?u=6474126&d=6474126&m=0&b=0&forceRefresh=1&matrix=1&debug=1>
LCPlaylist2

<http://209.67.158.18/servlet/rateme?action=RATING%5FEVENT&rater=222382&ratee=648416&rating=91&bucket%5Fnumber=100&ratee%5Ftype=1> LCPlaylist1
<http://209.67.158.29/servlet/rateme?action=RATING%5FEVENT&rater=222382&ratee=648416&rating=91&bucket%5Fnumber=100&ratee%5Ftype=1> LCPlaylist2

From: Boulter, Jeff
Sent: Friday, September 24, 1999 2:44 PM
To: Arango, Padgett
Cc: Beaupre, Todd
Subject: RE: links to radio stations

Eventually this page should just have real rating widgets on them instead of a link to rate the artists only.

Jeff

-----Original Message-----

From: Arango, Padgett
Sent: Friday, September 24, 1999 11:42 AM
To: Boulter, Jeff
Subject: RE: links to radio stations

someone seems to have put the whole page inside a <script> tag for reasons i fail to understand. try it now...

-----Original Message-----

From: Boulter, Jeff
Sent: Friday, September 24, 1999 11:31 AM
To: Arango, Padgett
Subject: RE: links to radio stations

When I use these on devweb6, I get a blank page. Example:

http://devweb6.launch.com/playlist/fs_Start/1,4489,1070,00.html

Jeff

-----Original Message-----

From: Arango, Padgett
Sent: Friday, September 24, 1999 11:24 AM
To: Plasecki, David; Boulter, Jeff
Subject: links to radio stations

Ideally, you would do the following:

[CURL /playlist/station stationID]

todd mentions that this may not work due to the cas wackiness. perhaps we should stage it, and, if it doesn't work, i can recode playlists so that it can take a querystring, so we don't have to deal with curts...

-p

From: Boulter, Jeff
Sent: Monday, September 27, 1999 12:22 PM
To: Sarver, Evan; Beaupre, Todd
Subject: RE: interface of launchcast errors

Yes, I know, the phrasing has been changed and the phrasing in the player interface has not been updated. It takes some programming work to get the "Your" vs. "Hitsman's" thing done.

Jeff

-----Original Message-----

From: Evan Sarver [mailto:evans@launch.com]
Sent: Sunday, September 26, 1999 5:07 PM
To: toddb@launch.com; jeffb@launch.com
Subject: interface of launchcast errors

The LaunchCast interface seems to have several linguistic errors.

For instance, it is now telling me that the reason it is playing the song is:
"Your rated this song highly."

Pretty good English! I have a writing degree, so I'm more nitpicky than the average user, but this should be either "your high rating for this song" or "you rated this song highly." I've noticed other spelling and grammatical errors before, though they may be taken care of by now.

Subject: RE: ads in launchcast player
Date: Thu, 30 Sep 1999 10:25:43 -0800
From: Boulter, Jeff
To: Martin, Andrew

Yup.

> -----Original Message-----
> From: Martin, Andrew
> Sent: Thursday, September 30, 1999 11:24 AM
> To: Boulter, Jeff
> Cc: Beaupre, Todd; Chang, Lee
> Subject: RE: ads in launchcast player

> Looks like these are pulling better today, yes?

> -----Original Message-----
> From: Boulter, Jeff
> Sent: Wednesday, September 29, 1999 1:57 PM
> To: Martin, Andrew
> Cc: Beaupre, Todd; Chang, Lee
> Subject: ads in launchcast player

> The ads in the launchcast player don't seem to be loading about
> 50% of the time. They just timeout after a while. It seems that they
> load immediately or not at all. The only thing different on our end is
> the ordinal used for the request, but there doesn't seem to be a pattern
> to the ordinal value and the ad loading or now.

> Here's a URL to test it directly:

> [http://devweb6.launch.com/music/launchcast/player/not_adframe/1,5169,,00](http://devweb6.launch.com/music/launchcast/player/not_adframe/1,5169,,00.html)
> .html

> Jeff

From: "Boulter, Jeff"
Sent: Friday, October 01, 1999 10:10 AM
To: "Beaupre, Todd"
Subject: your turn

Attachments: Playlist Generator Timings.xls

random songs is now cranked up to 5000

The URL I used is this: <http://devweb7/servlet/playlist?u=<userID>&debug=1&forceRefresh=1>

Let me know if you have any questions on where to get the numbers.



Playlist Generator
Timings.xls...

Jeff

From: "Boulter, Jeff"
Sent: Friday, October 01, 1999 2:00 PM
To: "Beaupre, Todd"
Subject: FW: player widget bug

Attachments: player.swf



player.swf (99 KB)

Was this your understanding regarding the horizontal widget?

Jeff

-----Original Message-----

From: Glenn Thomas - Smashing Ideas [mailto:glennt@smashingideas.com]
Sent: Friday, October 01, 1999 1:07 PM
To: Boulter, Jeff
Subject: Re: player widget bug

I haven't seen the vertical slider bug yet. The radio icon is gone.

I believe I've found the final place that changes the song name to orange from white and that should be fixed. It shouldn't cause any problems elsewhere, but I'll keep my eye on it.

The horizontal widget never had the capability to change where the button was on the slider. It's always resolved to the tens. This is based on the original design as well as the size it is placed at. There was a concern that the button did not have enough space in between the color rectangles to sit without being too cluttered looking.

Glenn
Smashing Ideas
www.smashingideas.com
206.378.0100

----- Original Message -----

From: Boulter, Jeff <jeffb@launch.com>
To: Glenn Thomas (E-mail) <glennt@smashingideas.com>
Sent: Friday, October 01, 1999 12:19 PM
Subject: player widget bug

>
> If I already have a rating for something (say 30) and I start moving the
> slider up, the number doesn't start changing until I reach 35, then it
> works normally.
>
> Jeff

From: "Boulter, Jeff"
Sent: Friday, October 01, 1999 3:46 PM
To: "Cheng, Cheng-Wei"
Subject: optimize?

Can you optimize this? We're having problems with it.

sp_lcGetPlayingInfoForUser_xsxx <userID>

Thanks,

Jeff

From: "Boulter, Jeff"
Sent: Friday, October 01, 1999 3:58 PM
To: 'Glenn Thomas - Smashing Ideas'
Subject: RE: player updating

No, all the info for the song - avg rating, rating, etc... It looks like you're still listening to the previous song.

Can we increase the timeout on this? Or just keep retrying?

Jeff

-----Original Message-----

From: Glenn Thomas - Smashing Ideas [mailto:glennt@smashingideas.com]
Sent: Friday, October 01, 1999 4:02 PM
To: Boulter, Jeff
Subject: Re: player updating

jeff,

is that the only thing not reloading?

glenn
Smashing Ideas
www.smashingideas.com
206.378.0100

----- Original Message -----

From: Boulter, Jeff <jeffb@launch.com>
To: Glenn Thomas (E-mail) <glennt@smashingideas.com>
Sent: Friday, October 01, 1999 3:49 PM
Subject: player updating

>
> We're seeing some problems with the song info updating on the beta site.
> Sometimes it never updates when you change songs (either normally or via
> skip). Do you have any suspicions on what the problem may be on your end?
> We're working on optimizing our page that returns the data in case that
> might be causing a problem.
>
> Thanks,
>
> Jeff

From: "Boulter, Jeff"
Sent: Sunday, October 03, 1999 2:56 PM
To: 'Glenn Thomas - Smashing Ideas'
Subject: RE: widget + player

Thanks Glenn.

I put them both in.

Something I noticed: It always says I'm listening to _AndreasgyT_'s station!
The relative ratings don't seem to work

I just got a song whose title did not show up at all. The song's name was "That's The Way I Feel" - a single quote issue?

I'm getting radio stations in my player, but they're pretty rare.

Other responses below...

-----Original Message-----

From: Glenn Thomas - Smashing Ideas [mailto:glennnt@smashingideas.com]
Sent: Sunday, October 03, 1999 2:42 PM
To: Boulter, Jeff
Subject: widget + player

jeff,

take a look at the widget and see if you like this functionality. here's my one question about it: currently if you rate a song "never play again" you can never get it back. do you want to change this up or is that the way you all want it to work.

Yes, that's how it should work.

we should double check on the widget that everything is being properly sent out. once you get it up let me know and i'll hit some artist pages to rate songs and then come back later to make sure the changes are captured.

I'll do that

the player has one minor change : the volume on the slider should now go all the way to 0, prior to this it still went to .1. i've changed that but need to hear it work off your site to make sure it's functioning.

Works great!

It's not working at all to use ICQ to send attachments to you any more - what's your AOL account? i'll try that if you want

I'm "Bowlah" on AOL.

glenn

From: Boulter, Jeff
Sent: Monday, October 04, 1999 8:43 PM
To: 'Glenn Thomas (E-mail)'
Subject: another idea

I found a correlation between some different problems:

When the name of my station does not appear at the top, I can't rate anything at all. None of the ratings get sent. When this happens, my relative ratings are also blank - it doesn't say "Rating 70" or anything on the mouseovers.

I hope that helps.

Jeff

From: Boulter, Jeff
Sent: Monday, October 04, 1999 9:12 PM
To: 'Glenn Thomas - Smashing Ideas'
Subject: RE: another idea

I take that back - the player doesn't seem to be getting the djName at all anymore. I still see it being passed in the HTML.

Jeff

-----Original Message-----

From: Glenn Thomas - Smashing Ideas [mailto:glenn@smashingideas.com]
Sent: Monday, October 04, 1999 6:05 PM
To: Boulter, Jeff
Subject: Re: another idea

That makes sense because the player doesn't have a dj ID to apply the rating to. Can you send me the HTML code that's pulling up the player for you? I'm guessing that the ID is not being passed properly for you - or else is the wrong one. It's also set up that the dj ID is separate from the rating ID, so in your case the rating ID is more important.

It sounds like neither is being passed to the Flash player though.

Glenn
Smashing Ideas
www.smashingideas.com
206.378.0100

----- Original Message -----

From: Boulter, Jeff <jeffb@launch.com>
To: Glenn Thomas (E-mail) <glenn@smashingideas.com>
Sent: Monday, October 04, 1999 5:43 PM
Subject: another idea

> I found a correlation between some different problems:

>
> When the name of my station does not appear at the top, I can't rate
> anything at all. None of the ratings get sent. When this happens, my
> relative ratings are also blank - it doesn't say "Rating 70" or anything
> on
> the mouseovers.

> I hope that helps.

> Jeff

> Jeff Boulter
> jeffb@launch.com
> Senior Director, Product Development
> Launch Media, Inc.
> www.launch.com
> 310-526-4387

From: Boulter, Jeff
Sent: Monday, October 04, 1999 9:09 PM
To: 'Glenn Thomas - Smashing Ideas'
Cc: Beaupre, Todd
Subject: RE: another idea

Actually, the djID is not for rating, the rater is for rating. The djID is only for displaying whose station you're listening to. This is very important because if you use the djID to rate things and you're listening to someone else's station, you're rating things for another user.

This latest played doesn't display the average rating or anything else when it doesn't get the station name. The strange thing is that it happens sometimes. Half the time I do get my station name. I still can't rate things though.

Here's a URL for the player part of the player window. This one has my userID in it:

http://devweb6.launch.com/music/launchcast/player/not_player/1.4046,,.PF.html?djID=6474126&bandwidth=0&moodID=0&random=3759

I did see rater, djID, and djName being passed properly to the player in the source of this page.

Jeff

-----Original Message-----

From: Glenn Thomas - Smashing Ideas (mailto:glenn@smashingideas.com)
Sent: Monday, October 04, 1999 6:05 PM
To: Boulter, Jeff
Subject: Re: another idea

That makes sense because the player doesn't have a dj ID to apply the rating to. Can you send me the HTML code that's pulling up the player for you? I'm guessing that the ID is not being passed properly for you - or else is the wrong one. It's also set up that the dj ID is separate from the rating ID, so in your case the rating ID is more important.

It sounds like neither is being passed to the Flash player though.

Glenn
Smashing Ideas
www.smashingideas.com
206.378.0100

----- Original Message -----

From: Boulter, Jeff <jeff@launch.com>
To: Glenn Thomas (E-mail) <glenn@smashingideas.com>
Sent: Monday, October 04, 1999 5:43 PM
Subject: another idea

>
> I found a correlation between some different problems:
>
> When the name of my station does not appear at the top, I can't rate
> anything at all. None of the ratings get sent. When this happens, my
> relative ratings are also blank - it doesn't say "Rating 70" or anything
> on
> the mouseovers.
>
> I hope that helps.
>
> Jeff
>
>
>

LMI 27387

From: "Boulter, Jeff"
Sent: Monday, October 04, 1999 11:41 AM
To: 'Glenn Thomas (E-mail)'
Subject: player widget bug

When you hit the X for an artist, it skips the song, but does not save the rating. I don't know if this is broken for albums too.

Thanks,

Jeff

From: Boulter, Jeff
Sent: Monday, October 04, 1999 11:56 PM
To: 'Glenn Thomas - Smashing Ideas'
Subject: RE: yet more little bugs

Yes, I'm noticing that the player is not using any of the data it's getting from the lookup_url except for the rating.

You can check it manually at
http://devweb6.launch.com/music/launchcast/player/not_getsonginfo/?rater=6474126

Jeff

From: "Boulter, Jeff"
Sent: Monday, October 04, 1999 11:55 AM
To: 'Glenn Thomas - Smashing Ideas'
Subject: RE: player widget bug

If you click the artist tab and rate it an X, it should skip and update the rating for that artist to 0. From then on, no songs will be played by that artist.

If you click the album tab and rate it an X, it should skip and update the rating for that album to 0. From then on, no songs will be played on that album.

Jeff

-----Original Message-----

From: Glenn Thomas - Smashing Ideas [mailto:glennt@smashingideas.com]
Sent: Monday, October 04, 1999 11:55 AM
To: Boulter, Jeff
Subject: Re: player widget bug

actually - how is it supposed to work : never play the album again, never play the artist again or is it just never play the song no matter what?

glenn
Smashing Ideas
www.smashingideas.com
206.378.0100

----- Original Message -----

From: Boulter, Jeff <jeffb@launch.com>
To: Glenn Thomas (E-mail) <glennt@smashingideas.com>
Sent: Monday, October 04, 1999 11:40 AM
Subject: player widget bug

>
> When you hit the X for an artist, it skips the song, but does not save
> the rating. I don't know if this is broken for albums too.
>
> Thanks,
>
> Jeff

From: "Boulter, Jeff"
Sent: Monday, October 04, 1999 5:43 PM
To: 'Glenn Thomas (E-mail)'
Subject: another idea

I found a correlation between some different problems:

When the name of my station does not appear at the top, I can't rate anything at all. None of the ratings get sent. When this happens, my relative ratings are also blank - it doesn't say "Rating 70" or anything on the mouseovers.

I hope that helps.

Jeff

From: Boulter, Jeff
Sent: Tuesday, October 05, 1999 5:50 PM
To: 'Glenn Thomas - Smashing Ideas'
Subject: RE: getting close

It's in.

-----Original Message-----

From: Glenn Thomas - Smashing Ideas [mailto:glenn@smashingideas.com]
Sent: Tuesday, October 05, 1999 2:47 PM
To: Boulter, Jeff
Subject: getting close

jeff,

another file with a piece of code to try and bug trap what's going on with the X on the widget. It should now tell us exactly what is going out as the rating.

glenn
Smashing Ideas
www.smashingideas.com
206.378.0100

From: Boulter, Jeff
Sent: Wednesday, October 06, 1999 3:30 PM
To: 'Glenn Thomas (E-mail)'
Subject: FW: Bug while rating...

This is it.

-----Original Message-----

From: Beaupre, Todd
Sent: Wednesday, October 06, 1999 12:29 PM
To: Boulter, Jeff
Subject: FW: Bug while rating...

ha ha

-----Original Message-----

From: BrianM [mailto:BrianM@launch.com]
Sent: Wednesday, October 06, 1999 12:41 PM
To: toddb@launch.com
Subject: Bug while rating...

Todd:

I was going through and rating all of the Beastie Boys albums and noticed that on the albums:

The Sound From Way Out
Get It Together (Maxi Single)
Ill Communication
So Watcha Want EP

That when I went to rate it I can only rate it a 100. No matter what other rating I try to give it the ratings widget shoots up to 100. It's a great ploy by their PR person but I don't necessarily want them to be rated that way.

Thanks,
Brian

From: "Boulter, Jeff"
Sent: Wednesday, October 06, 1999 12:30 PM
To: 'Glenn Thomas (E-mail)'
Subject: FW: Bug while rating...

This is it.

-----Original Message-----

From: Beaupre, Todd
Sent: Wednesday, October 06, 1999 12:29 PM
To: Boulter, Jeff
Subject: FW: Bug while rating...

ha ha

-----Original Message-----

From: BrianM [mailto:BrianM@launch.com]
Sent: Wednesday, October 06, 1999 12:41 PM
To: toddb@launch.com
Subject: Bug while rating...

Todd:

I was going through and rating all of the Beastie Boys albums and noticed that on the albums:

The Sound From Way Out
Get It Together (Maxi Single)
Ill Communication
So Watcha Want EP

That when I went to rate it I can only rate it a 100. No matter what other rating I try to give it the ratings widget shoots up to 100. It's a great ploy by their PR person but I don't necessarily want them to be rated that way.

Thanks,
Brian

From: "Boulter, Jeff"
Sent: Wednesday, October 06, 1999 1:32 PM
To: "From, Linda"
Subject: ratings bug

Can you reliably reproduce that bug where the rating widget was stuck on 100? I think it was on the top 100 page.

Jeff

From: "Boulter, Jeff"
Sent: Wednesday, October 06, 1999 2:31 PM
To: 'Glenn Thomas (E-mail)'
Subject: FW: ratings bug

Another to investigate, possibly related?

Jeff

-----Original Message-----

From: From, Linda
Sent: Wednesday, October 06, 1999 2:32 PM
To: Boulter, Jeff
Subject: RE: ratings bug

No, I can't reproduce it. It has happened to me four times so far.

One thing that I noticed is the ratings that are being set to 100 are not being saved. If I click the next>> link, and then use the <<previous link to return to the page, those songs that were being set to 100 don't have a rating.

-Linda

-----Original Message-----

From: Boulter, Jeff
Sent: Wednesday, October 06, 1999 1:32 PM
To: From, Linda
Subject: ratings bug

Can you reliably reproduce that bug where the rating widget was stuck on 100? I think it was on the top 100 page.

Jeff

From: "Boulter, Jeff"
Sent: Wednesday, October 06, 1999 4:24 PM
To: 'Glenn Thomas (E-mail)'
Cc: "Beaupre, Todd"
Subject: the latest list

1. Get AOL Instant Messenger working. I'm "Bowtah". I think it will save us a lot of time. AOL 3.0 also has the ability to send files. I miss ICQ.
2. player - The "new!" appears outside the player frame for songs with long titles
2. player - "jiggling" of the song, album, and artist titles
4. player - dragging on buttons
5. player - the relative rating at 100 appears over the rating
6. player - make the relative rating mouseovers appear while the mouse is down
7. player - make the mouseovers for pause, skip, X, etc appear sooner. The pause right now seems like at least a second.
8. widget - widget stuck on 100 (hard to reproduce)
9. widget - when you rate something a 90, the 100 button lights up blue
10. widget - 100 ratings are not being saved (can't reproduce)
11. widget - If you click in the white above the X, it will rate it a 50.
12. widget - When you rate something an X, all the buttons turn black

Jeff Boulter

From: "Boulter, Jeff"
Sent: Thursday, October 07, 1999 3:56 PM
To: 'Glenn Thomas (E-mail)'
Subject: more issues

15. player - when personalizing, then notified of the start of play, change personalizing to updating

16. player - really nitpicky thing: the updating and personalizing seem to have a line on the right side bottom half edge

17. player - a song came up that did not have a song title. The title of the song was "It's Too Late" by the Johnny Rivers/L.A. Boogie Band. Could it be the single quote? I checked the song info and it was escaped properly?

18. The Why Played mouseover appears in strange situations. I used to be able to reproduce it by skipping, then quickly activating another window. It always seems to appear when the player window is not the active window.

Is this a good way for you to work? Would you rather I send these issues to you in some other format? What would be really cool would be if there was some sort of web-based issue tracking system. Someday...

Jeff

From: Boultar, Jeff
Sent: Thursday, October 07, 1999 7:24 PM
To: Beaupre, Todd
Subject: RE: ratings issue

Cached pages on the client? I added some anti-cache tags on some of the pages on the dev server already.

Jeff

From: "Boulter, Jeff"
Sent: Friday, October 08, 1999 11:48 AM
To: 'Glenn Thomas - Smashing Ideas'
Subject: RE: widget

Thanks glenn. I put it up there, but I doubt you'll be able to get to it because they changed all our IPs last night for the new firewall. Can you send me your IP again so I can unblock it for you?

Can you implement the "unrate" thing using clicking on the same rating twice? For example, if you already have a rating for something, clicking on that rating again will delete the rating.

For #17, I'm 100% sure that it was escaped properly. It looked like this in the song_info:

song_name=It's Too Late

Some more issues that came up last night:

19. player - clicking anywhere on the player when the player is not the active window shouldn't do anything but make the player window active

20. player - the area in the very corners of the player, outside the rounded edges, should be black

21. player - sometimes it takes a long time to update the song info. We need to start showing the updating movie if it takes more than a second or two. I'm in the process of writing the song lookup page in Java, which should make it faster.

Jeff

-----Original Message-----

From: Glenn Thomas - Smashing Ideas [mailto:glennt@smashingideas.com]
Sent: Friday, October 08, 1999 11:38 AM
To: Boulter, Jeff
Subject: Re: widget

Jeff,

I've attached both a player file. The last widget file I sent should have incorporated all fixes.

I will be away this weekend but will give some thought to the drag over problem with relative ratings. I'll get to work on the buttons drag over on Monday. With regard to the Song Title problem, could you please send me the song_lookup_url so that I can access the information that is being returned and run it through the code step by step. That should clear up this little mystery in the quickest possible manner.

Glenn

> 2. player - The "new!" appears outside the player frame for songs with
long
> titles

DONE

> 3. player - "jiggling" of the song. album, and artist titles

DONE

> 4. player - dragging on buttons - You should be able click on a widget

> button in the player and drag up and down to change the value of the rating
> in increments of ten

NOT DONE

> 6. player - make the relative rating mouseovers appear while the mouse is down

NOT DONE

> 7. player - make the mouseovers for pause, skip, X, etc appear sooner.
> The pause right now seems like at least a second.

DONE

> 8. widget - widget stuck on 100 (hard to reproduce)

OUTSTANDING ISSUE

> 9. widget - when you rate something a 90, the 100 button lights up blue

DONE

> 10. widget - 100 ratings are not being saved (can't reproduce)

OUTSTANDING ISSUE

> 11. widget - If you click in the white above the X, it will rate it a 50.

DONE

> 12. widget - When you rate something an X, all the buttons turn black

DONE

> 13. widget & player - We need a way to 'unrate' a song. I'm not sure the best interface to do this. You could just click on the same value again. In this case you would send -1 as the rating and we would delete it from the database. If you have other interface ideas, let me know.

NEED INTERFACE IDEA - DISCUSS

> 14. player - Mouseovers for song, album, and artist should show the full title.

DONE

> 15. player - when personalizing, then notified of the start of play, change personalizing to updating

DONE

16. player - really nitpicky thing: the updating and personalizing seem to have a line on the right side bottom half edge

DONE - good eyes

17. player - a song came up that did not have a song title. The title of the song was "It's Too Late" by the Johnny Rivers/L.A. Boogie Band. Could it be the single quote? I checked the song info and it was escaped properly?

OUTSTANDING ISSUE

18. The Why Played mouseover appears in strange situations. I used to be able to reproduce it by skipping, then quickly activating another window. It always seems to appear when the player window is not the active window.

DONE

From: "Boulter, Jeff"
Sent: Friday, October 08, 1999 1:02 PM
To: "Bosick, Tom"
Cc: "Lee, Howard"
Subject: devweb needs

We need to have devweb2, 6, and 7 resolvable internally.

Additionally, our flash developers in Seattle need to be able to access these servers. Their subnet is: 216.160.66.*

Jeff

From: Glenn Thomas - Smashing Ideas [mailto:glenn@smashingideas.com]
Sent: Friday, October 08, 1999 1:06 PM
To: Boulter, Jeff
Subject: Re: widget

IP
216.160.66.115 and 216.160.66.116

17. Escaped song name. Right, but I need to get the song_lookup_url in the html code so I can force that song to come up on my computer and then go through it step by step how it is pulling it into Flash and displaying it.

Andreas has code that makes the song name empty if certain requirements are met. The song name with the escaped characters shouldn't be meeting those requirements, but it must be in some way. I need to find out what that way is and the only way for me to do that is to see how the song name comes in and goes through the code.

> 19. player - clicking anywhere on the player when the player is not
> the active window shouldn't do anything but make the player window
> active

this would have to be done in concert with javascript. can you create a function that will tell that window when it is inactive? if you can do that, then we can make it run when the window is inactive and tell the Flash player to make the other buttons, etc. inactive until it is active again. we would also then need a javascript function that tells the flash player that the window is active.

> 20. player - the area in the very corners of the player, outside the
rounded
> edges, should be black

No problem

> 21. player - sometimes it takes a long time to update the song info.
> We
need
> to start showing the updating movie if it takes more than a second or
two.
> I'm in the process of writing the song lookup page in Java, which
> should make it faster.

Is the song audio already playing when this happens for you?

Glenn

From: "Boulter, Jeff"
Sent: Monday, October 11, 1999 10:24 AM
To: 'Glenn Thomas - Smashing Ideas'
Cc: "Beaupre, Todd"
Subject: RE: widget

Glenn - updated list below with my comments.

The songinfo on the dev site is now being returned from a java servlet - should be faster.

Jeff

4. player - dragging on buttons
6. player - make the relative rating mouseovers appear while the mouse is down
8. widget - widget stuck on 100 (hard to reproduce)
10. widget - 100 ratings are not being saved (can't reproduce)
13. widget & player - We need a way to 'unrate' a song. I'm not sure the best interface to do this. You could just click on the same value again. In this case you would send -1 as the rating and we would delete it from the database. If you have other interface ideas, let me know.
14. player - Mouseovers for song, album, and artist should show the full title.
16. player - really nitpicky thing: the updating and personalizing seem to have a line on the right side bottom half edge - I'm still seeing this.
17. player - a song came up that did not have a song title. The title of the song was "It's Too Late" by the Johnny Rivers/L.A. Boogie Band. Could it be the single quote? I checked the song info and it was escaped properly.

media_id=41639&song_id=486171&song_name=It%27s+Too+Late&album_id=54029
&album_name=Anthology+1964-77+%281991%29&artist_id=1030483&artist_name=Johnny+Rivers%
2FL.A.+Boogie+Band&exclusive=0&comm_rating=48&new=0&origin=You+rate+d+this+song&popular=0
&song_rating=100&song_rating_type=1&album_rating=-1&album_rating_type=1&artist_rating=-1
&artist_rating_type=1&fan_name=boulter&fan_id=6474126&fan_online=1
&ticker_text=&image_url=
19. player - clicking anywhere on the player when the player is not the active window shouldn't do anything but make the player window active. I'll work on a JavaScript function to notify the player when it is in/active.
20. player - the area in the very corners of the player, outside the rounded edges, should be black
21. player - sometimes it takes a long time to update the song info. We need to start showing the updating movie if it takes more than a second or two. The audio is already playing.
22. player - we lost the mouseover for 90+ that says Rating 90 - Favorite! - play most frequently
23. player & widget - show status of saved or not. Something on the player should indicate when a rating is being saved. Maybe a little arrow flashes somewhere?

From: "Boulter, Jeff"
Sent: Monday, October 11, 1999 11:30 AM
To: "Lee, Howard"
Subject: new servlet

we're now returning the song info to the player via a servlet instead of a storyserver page. This servlet will live on the lcPlaylist boxes and is called GetSongInfoServlet. It should be aliased as songinfo. It's already setup on dewweb7. The same configuration should be on lcplaylist the next time you prop.

Also, the media gateway should also be running on lcplaylist from now on (the next time we prop). The lcstream boxes will only serve windows media. Jrun should not be running on them at all. I've updated the constants to reflect this.

Jeff

From: "Boulter, Jeff"
Sent: Monday, October 11, 1999 1:34 PM
To: "Beaupre, Todd"

```
public byte origin()
{
    double maxValue = 0;
    byte maxSource = SOURCE_RANDOM;
    byte ratingSource = 0;

    if (rating.IsSet())
        ratingSource = rating.getSource();

    if (info.commRating > maxValue && info.commRating > POPULAR_THRESHOLD && ratingSource != 1)
    {
        maxValue = info.commRating;
        maxSource = SOURCE_POPULAR;
    }

    if (djsAverage.IsSet() && djsAverage.get() >= maxValue && djsAverage.get() > 0 && ratingSource != 1)
    {
        maxValue = djsAverage.get();
        maxSource = SOURCE_DJS;
    }

    /*
    if (netP > maxValue)
    {
        maxValue = netP;
        maxSource = SOURCE_NETP;
    }
    */

    if (bds > 0 && bds >= maxValue && ratingSource != 1)
    {
        maxValue = bds;
        maxSource = SOURCE_BDS;
    }

    // according to the weight matrix, if there's an explicit rating,
    // that's the only source
    // but let's lie to people because they don't like it when we say
    // we played lowly-rated songs for them
    // even though that's what we say we will play anyway

    if (rating.IsSet())
    {
        short value = rating.get();

        if (value > MIN_RATING_FOR_RATED_SOURCE && value >= maxValue)
        {
            maxValue = value;
        }
    }
}
```



```
        maxSource = ratingSource;
    }
}
// lies, lies, lies.
if (maxValue < MIN_RATING_FOR_RATED_SOURCE)
{
    maxSource = SOURCE_RANDOM;
}
return maxSource;
}
```

Jeff Boulter

From: "Boulter, Jeff"
Sent: Monday, October 11, 1999 1:51 PM
To: 'Glenn Thomas - Smashing Ideas'
Cc: "From, Linda"
Subject: RE: widget

yeah!

-----Original Message-----

From: Glenn Thomas - Smashing Ideas [mailto:glennt@smashingideas.com]
Sent: Monday, October 11, 1999 1:53 PM
To: Boulter, Jeff
Subject: Re: widget

8. widget - widget stuck on 100 (hard to reproduce)

I finally reproduced this and should be able to track it down and fix it.

Glenn
Smashing Ideas

From: "Boulter, Jeff"
Sent: Tuesday, October 12, 1999 11:51 AM
To: 'Glenn Thomas (E-mail)'
Subject: FW: Strange error with Flash and LaunchCast

Attachments: launchcastbug.jpg



launchcastbug.jpg
(87 KB)

We've seen this error with windows media player before, so I think it's actually an IE problem. I think it happens when the user does not have administrator priviledges on their box and cannot install software.

If you have any ideas that lead to the contrary, please let me know.

Jeff

-----Original Message-----

From: Evan Sarver [mailto:evans@launch.com]
Sent: Tuesday, October 12, 1999 11:50 AM
To: Jeff Boulter (E-mail); Todd Beaupre (E-mail)
Subject: Strange error with Flash and LaunchCast

Hey guys,

I just tried to load launchcast, and got the error message in the JPG attached.

Evan

From: "Boulter, Jeff"
Sent: Tuesday, October 12, 1999 6:39 PM
To: "Marshall, Brian"; "Beaupre, Todd"
Cc: "Boulter, Jeff"
Subject: RE: Where's My LaunchCast (quickly suffering withdraws)

We're upgrading with bug fixes, etc. Will be back shortly.

Jeff

-----Original Message-----

From: BrianM [mailto:BrianM@launch.com]
Sent: Tuesday, October 12, 1999 6:49 PM
To: toddb@launch.com
Cc: jeffb@launch.com
Subject: Where's My LaunchCast (quickly suffering withdraws)

From: "Boulter, Jeff"
Sent: Wednesday, October 13, 1999 10:39 AM
To: "Lee, Howard"
Subject: RE: URL's

can you update this with all the new servlets and servers? Better yet, put it on a web page on jennings.

Thanks,

Jeff

-----Original Message-----

From: Lee, Howard
Sent: Wednesday, September 22, 1999 8:28 PM
To: Boulter, Jeff; Beaupre, Todd
Subject: FW: URL's

These are test URL's based on the servlets and the params we discussed last night. I dunno if this email is extraneous but hey it might be convenient who knows.

-h-

-----Original Message-----

From: Lee, Howard
Sent: Wednesday, September 22, 1999 7:51 PM
To: Lee, Howard
Subject: URL's

<http://209.67.158.17/servlet/gateway?u=6474126&debug=1> LaunchStream1
<http://209.67.158.16/servlet/gateway?u=6474126&debug=1> LaunchStream2

<http://209.67.158.18/servlet/playlist?u=6474126&d=6474126&m=0&b=0&forceRefresh=1&matrix=1&debug=1>
LCPlaylist1
<http://209.67.158.29/servlet/playlist?u=6474126&d=6474126&m=0&b=0&forceRefresh=1&matrix=1&debug=1>
LCPlaylist2

<http://209.67.158.18/servlet/rateme?action=RATING%5FEVENT&rater=222382&ratee=648416&rating=91&bucket%5Fnumber=100&ratee%5Ftype=1> LCPlaylist1
<http://209.67.158.29/servlet/rateme?action=RATING%5FEVENT&rater=222382&ratee=648416&rating=91&bucket%5Fnumber=100&ratee%5Ftype=1> LCPlaylist2

From: Boulter, Jeff
Sent: Wednesday, October 13, 1999 10:55 PM
To: 'Glenn Thomas (E-mail)'
Subject: latest issues

A few more came up today.

Flash Player issues

13. widget & player - We need a way to 'unrate' a song. I'm not sure the best interface to do this. You could just click on the same value again. In this case you would send -1 as the rating and we would delete it from the database. If you have other interface ideas, let me know.

14. player - Mouseovers for song, album, and artist should show the full title.

16. player - really nitpicky thing: the updating and personalizing seem to have a line on the right side bottom half edge - I'm still seeing this.

19. player - clicking anywhere on the player when the player is not the active window shouldn't do anything but make the player window active. I'll work on a JavaScript function to notify the player when it is in/active.

21. player - sometimes it takes a long time to update the song info. We need to start showing the updating movie if it takes more than a second or two

23. widget - show status of saved or not. Something on the player should indicate when a rating is being saved. Maybe a little arrow flashes somewhere?

24. player - put the DJ ratings before the fans

25. player - the hand cursor appears on the why played area even though it isn't clickable

26. player - various artists, original soundtrack shouldn't jiggle

27. player - rating mouseovers should say rating X - Never play again (same for album, and artist)

28. player - Your DJs' Ratings -> Your DJs' Song Ratings

29. player - I witnessed one user for whom the player would not load at all in Netscape. All it would do is remain white. Right clicking on the white area showed that flash 4 was installed, but the movie was not loaded.

30. spinning logo - for netscape users, it doesn't start spinning until you refresh the page

31. player - when dragging on the slider, it only shows relative ratings with values that are multiples of 10

Jeff Boulter
jeffb@launch.com
Senior Director, Product Development
Launch Media, Inc.
www.launch.com
310-528-4387

From: Boulter, Jeff
Sent: Thursday, October 14, 1999 6:27 PM
To: Beaupre, Todd
Cc: Kanfi, Yaniv
Subject: FW: LaunchCast request

Can you do this in access?

~~-----Original Message-----~~

From: Kanfi, Yaniv
Sent: Thursday, October 14, 1999 3:15 PM
To: Boulter, Jeff
Subject: LaunchCast request

Is there a URL that I can type and view all my song ratings (I want to import the data into a spreadsheet)? - don't ask why.

Yaniv.

From: "Boulter, Jeff"
Sent: Friday, October 15, 1999 2:24 PM
To: "Cheng, Cheng-Wei"
Subject: optimization

Do you think you could take all of these stored procs and create a new stored proc that returns all the data in one query? If you don't think that would be faster, that's fine.

```
sp_lcGetFeaturedDJInfo_xsx 13302
GO
sp_lcGetRatingsCountsForUser_xsx 13302
GO
sp_lcGetDJFanCount_xsx 13302
GO
sp_lcGetDJListenerCount_xsx 13302
```

Thanks,

Jeff

From: "Boulter, Jeff"
Sent: Friday, October 15, 1999 2:56 PM
To: "Cheng, Cheng-Wei"
Subject: one more to add

Can you integrate this stored proc too?

sp_lcGetPlayingForUser_xsxx

example userIDs:

6474126

0

13302

Thanks!

Jeff

From: "Boulter, Jeff"
Sent: Friday, October 15, 1999 5:36 PM
To: 'Glenn Thomas (E-mail)'
Subject: problem with player

The latest version of the player seems to be doing weird things when I click on the rating buttons. When I click to rate something, it will jump back to the previous rating (or 50 if none). I have to click several times for it to stay at a rating value.

Jeff

From: Boulter, Jeff
Sent: Friday, October 15, 1999 6:01 PM
To: 'Glenn Thomas - Smashing Ideas'
Subject: RE: latest issues

various artists IDs are:

ARTIST_VARIOUS_ARTISTS = 1028125;
ARTIST_ORIGINAL_SOUNDTRACK - 1020156;
ARTIST_SOUNDTRACK = 1036715;

You should already have this somewhere in the code though, because you dont allow clicking on Various Artists or rating of them.

Jeff

-----Original Message-----

From: Glenn Thomas - Smashing Ideas [mailto:glennt@smashingideas.com]
Sent: Thursday, October 14, 1999 8:20 AM
To: Boulter, Jeff
Subject: Re: latest issues

Jeff,

This sounds great. I'll definitely look forward to it going fully live.

I'll move forward on the remaining issues. I'm trying out a couple of different things on both the Unrating issue and getting the whole song, album and artist name to show. The problem here is that the length is so variable that it's a bit more complicated to create a routine that will show the name properly and not cover up the "jiggle" on the rollover.

I'll deliver a file tomorrow but here is what's been completely done and then questions about some of the requests:

> 16. player - really nitpicky thing: the updating and personalizing seem to
> have a line on the right side bottom half edge - I'm still seeing this.

FIXED - finally

> 23. widget - show status of saved or not. Something on the player should
> indicate when a rating is being saved. Maybe a little arrow flashes
> somewhere?

FIXED

> 25. player - the hand cursor appears on the why played area even though it
> isn't clickable

FIXED - we will need to test this because it might be a problem again when you quickly leave the browser window. The popup might come up in funky ways. It shouldn't but watch for it.

> 26. player - various artists shouldn't jiggle

What is the various artists number? I need to force that through so I can test a possible solution.

Glenn
Smashing Ideas

From: Glenn Thomas - Smashing Ideas [mailto:glennt@smashingideas.com]
Sent: Monday, October 18, 1999 9:40 AM
To: Boulter, Jeff
Subject: player

jeff,

this player now sends out a -1 to unrate, but it doesn't seem to accept it on the server/database side. give it a try, but from here it just sits on "rating..." and never comes back rated/unrated.

the two main issues outstanding for me are mouseovers showing the whole song, album, etc. and if updating takes a long time.

I'm looking into the netscape issue. I haven't seen anything yet.

Glenn
Smashing Ideas

From: "Boulter, Jeff"
Sent: Monday, October 18, 1999 2:52 PM
To: 'Glenn Thomas - Smashing Ideas'
Subject: RE: player

I would guess this is a player issue. If you can rate something, then unrate it, the servlet is working correctly. It seems like the player is saving some state or something which prevents it from unrating a second time.

Jeff

-----Original Message-----

From: Glenn Thomas - Smashing Ideas [mailto:glennt@smashingideas.com]
Sent: Monday, October 18, 1999 2:44 PM
To: Boulter, Jeff
Subject: Re: player

Jeff,

Is this a servlet issue then or a Flash issue for me to look into?

Glenn
Smashing Ideas

From: "Boulter, Jeff"
Sent: Monday, October 18, 1999 8:36 PM
To: "Hughes, Jim"; "Beaupre, Todd"
Cc: "Mico, Ted"; "Gorla, Peter"; "Fisher, Roberto"; "DiMartino, Dave"
Subject: RE: genres/LAUNCHcast DJ's

Yes, we have been encoding all genres. We have plenty (too much?) country music!

Once we get some stations with more diverse tastes in the system, we will put them in the list of DJs in the wizard. The way, someone could create a "country" station without being entirely forced into hearing music from only that genre.

We could make them up ourselves, but I think someone who's really into specific genres (like enlightenedone one and Sneetch) do a much better job.

Jeff

-----Original Message-----

From: Hughes, Jim
Sent: Monday, October 18, 1999 7:53 PM
To: Beaupre, Todd; Boulter, Jeff
Cc: Mico, Ted; Gorla, Peter; Fisher, Roberto; DiMartino, Dave
Subject: genres/LAUNCHcast DJ's

guys:

a) have we been encoding for all genres (do we have country, dance, etc. in there)?

b) i think in the wizard, we need to be offering a means for people to start listening right away, but with some genre-selecting capabilities...how about creating some users whose stations are completely genre-specific...

"or, if you'd prefer to start listening right away to a particular type of music check out the top DJ in each of the following genres:

Jazz daved
Techno wackyguy
"

etc. etc.

that way, if people ARE interested in having something programmed for them (read: no work) we're still promoting that by subscribing to another DJ is the way to go...

we could make these up ourselves, or get someone from the company for each of 'em...
thoughts?
jim

From: Boulter, Jeff
Sent: Monday, October 18, 1999 10:16 PM
To: 'Glenn Thomas (E-mail)'
Subject: latest issues

We're getting there!

Jeff

Flash Player issues

13. widget & player - We need a way to 'unrate' a song. I'm not sure the best interface to do this. You could just click on the same value again. In this case you would send -1 as the rating and we would delete it from the database. If you have other interface ideas, let me know. This is now working once, but once you rate something, then unrate it, you can no longer unrate it. I verified that it is NOT sending out the rating if you try to unrate something after you've unrated it once.

14. player - Mouseovers for song, album, and artist should show the full title.

19. player - clicking anywhere on the player when the player is not the active window shouldn't do anything but make the player window active. I'll work on a JavaScript function to notify the player when it is in/active.

21. player - sometimes it takes a long time to update the song info. We need to start showing the updating movie if it takes more than a second or two

23. widget - show status of saved or not. Something on the player should indicate when a rating is being saved. Maybe a little arrow flashes somewhere?

29. player - I witnessed one user for whom the player would not load at all in Netscape. All it would do is remain white. Right clicking on the white area showed that flash 4 was installed, but the movie was not loaded.

32. player - the widget tabs should be longer so that the album tab reaches the bottom of the widget area (missing about 1/8 inch at the bottom)

35. player - clicking the X button for an artist, album, or song does not skip and does not show the rating as saved.

36. player - skip is always sending mediaID 0

Jeff Boulter
jeffb@launch.com
Senior Director, Product Development
Launch Media, Inc.
www.launch.com
310-526-4387

From: Boulter, Jeff
Sent: Wednesday, October 20, 1999 6:11 PM
To: 'Glenn Thomas - Smashing Ideas'
Subject: RE: latest issues

What I meant is that you generate a random number and append it to the outgoing URL. The URL I pass to you is the same.

You'll just append it like this

/servlet/songinfo?rater=13302&volume=50&random=12345665

Are you saying that you can't generate a random number and append it this way? What about a date in a number format?

Naviscope's caching was causing this problem because when you request the same URL twice, it takes the cached version the second time and doesn't make the request to the servlet.

In the unrating case, the URL was the same both times. This was also preventing me from rating something an 80, rating it something else, then trying to rate it an 80 again. It would never make the request to save the 80 rating the second time.

Jeff

-----Original Message-----

From: Glenn Thomas - Smashing Ideas [mailto:glennt@smashingideas.com]
Sent: Wednesday, October 20, 1999 2:43 PM
To: Boulter, Jeff
Subject: Re: latest issues

Jeff,

I thought you meant that you were adding the randomly generated number from the servlet so I got returned

/servlet/songinfo?rater=...&volume=...&random=12345665

I don't think I can attach the random number in a meaningful way from within the Flash player. If I attach it the url comes out as:

/servlet/songinfo&random=12345665?rater=...&volume=...

This doesn't return anything at all.

> > 37. player - add a randomly generated number to the songinfo URLs so they
> > are not cached, like &random=8908055. The servlet will ignore this. The
> same
> > for the rateURLs in the player and widget.

With regard to the unrating - I can see the unrating value coming out in Naviscope as rating=t2D1 but then no confirmation is sent back by the servlet. Do you think this is part of the proxy problem or is it something else?

Glenn
Smashing Ideas
www.smashingideas.com

From: Boulter, Jeff
Sent: Wednesday, October 20, 1999 6:49 PM
To: Lee, Howard
Subject: stored procs

sp_lcSavePlaylist int @userID, image @playlist

delete from a200UserPlaylist

insert into it

sp_lcGetPlaylist int @userID

get playlist from a200UserPlaylist

Jeff Boulter
jeffb@launch.com
Senior Director, Product Development
Launch Media, Inc.
www.launch.com
310-526-4387

From: Glenn Thomas - Smashing Ideas [mailto:glennt@smashingideas.com]
Sent: Thursday, October 21, 1999 1:35 PM
To: Boulter, Jeff
Subject: player

Jeff,

I've gone through most of the changes. If you could test this one for the updating movie, I'd appreciate it. It should come up after about two seconds of no loading but since I can't check it in a real environment from here, this is all theory.

Unrating now has the random numbers at the end.

Skip now sends a mediaID

The X button now works.

There are now arrows on the rating text until it is saved.

Widget tabs are longer.

Did you get any javascript done to notify the player on active/inactive?

Still testing the best way to display long titles. How do you feel about a title scrolling across when you rollover it? I think I could do it so we still keep the jitter.

Glenn
Smashing Ideas
www.smashingideas.com
206.378.0100

From: Boulter, Jeff
Sent: Thursday, October 21, 1999 6:22 PM
To: 'Glenn Thomas - Smashing Ideas'
Subject: RE: player

Thanks Glenn.

I put both the latest player and widget in there. I'm surprised you can't get to devweb6. I can get to it from home. devweb2 does not work however. I'll ask some people here why you might be having problems.

I haven't see then the updating movie play after two seconds yet. Maybe you could crank it down to a half second, just for testing purposes.

Here's the latest and updated issues. There's a new few bugs with the latest player and widget.

Thanks,

Jeff

From: Boulter, Jeff
Sent: Friday, October 22, 1999 7:25 PM
To: Goetz, Richard; O'Darin, Stevie
Subject: RE: launchcast bugs

This has been fixed in the patch that will go out today or Monday.

Jeff

From: Cheng, Cheng-Wei
Sent: Monday, October 25, 1999 12:32 PM
To: Boultter, Jeff
Subject: list of table views and store proc and keys owned by dbClient

sp_lcUpdateWizardStepID_isux
sp_DigitalDownloadArtist_update
sp_lcGetUserDJRatingsForArtistID_xxxx
sp_lcGetUserDJRatingsForAlbumID_xxxx
sp_lcLogPlayNews_isud
sp_lcSavePlaylist_ixxd
sp_lcGetPlaylist_xxxx
sp_lcGetSongHistoryText_xxxx
sp_lcLogPlaySong_isud
sp_lcLogPlayAd_isud
sp_lcGetUserChosenGenreNames_xxxx
sp_lcGetWizardStepID_xxxx
sp_lcUpdatePlayHistoryText_isux
sp_lcSavePlayHistoryText_isux
sp_lcGetMediaPath_xxxx
sp_lcLogPlayTip_isud
sp_lcUpdatePopularDJs_ixxd
sp_DigitalDownloadArtist_delete
sp_lcGetUserSongTrace_xxxx
sp_lcGetUserSongDJTrace_xxxx

PK_a200UserPlayedArtistsCache
PK_a200UserSongHistoryText
PK_a200PopularDJs

a200UserArtistsPlayedCache
a200UserSongHistoryText
a200PopularDJ

v_mvVideos
temp va200SongGenres_ByUser
VIEW1

From: Boulter, Jeff
Sent: Monday, October 25, 1999 3:31 PM
To: Cheng, Cheng-Wei
Subject: RE: list of table views and store proc and keys owned by dbClient

Go ahead and change the ownership of all the stored procs that start with sp_lc and these tables:

a200UserArtistsPlayedCache
a200UserSongHistoryText
a200PopularDJ

Thanks,

Jeff

From: Beaupre, Todd
Sent: Monday, October 25, 1999 5:24 PM
To: Goria, Peter
Cc: Boutier, Jeff
Subject: RE: stats

62,092 song ratings (100%)
33,202 non-zero song ratings end in 0 (53%)
15,543 song ratings not ending in 0 (25%)
13,347 song ratings are 0 (21%)

-----Original Message-----
From: Goria, Peter
Sent: Monday, October 25, 1999 2:21 PM
To: Beaupre, Todd
Subject: stats
Importance: Low

Todd,

do you know what % of the ratings are precise (i.e. they don't end in 0)?

may be useful info from a UI design perspective.

thanks,
peter.

—
Peter Goria, VP Product Development
perezg@launch.com | LAUNCH.com | direct: 310 526 4325

From: Beaupre, Todd
Sent: Tuesday, October 26, 1999 9:30 AM
To: Goria, Peter
Cc: Boulter, Jeff
Subject: RE: new nav?

Nope. Shall we just throw it in on the dev site and see what happens?

-----Original Message-----

From: Goria, Peter
Sent: Tuesday, October 26, 1999 9:29 AM
To: Beaupre, Todd
Subject: new nav?

Have you guys taken a crack at the new navbar in LAUNCHcast?

Peter Goria, VP Product Development
peterg@launch.com | LAUNCH.com | direct: 310 526 4388

From: Boulter, Jeff
Sent: Wednesday, October 27, 1999 11:59 AM
To: Fritz, Cort; Beaupre, Todd
Subject: RE: LAUNCH/LAUNCHcast

We don't write any persistent cookies. All of our cookies are session-based and are set to ~~expire at the end of the session~~. In any case, the login and registration pages (not ours) set the user cookies. We don't touch them.

Jeff

From: Goria, Peter
Sent: Thursday, October 28, 1999 3:01 PM
To: Sichel, Jon
Subject: FW: Inventor's Disclosure Short Form for LAUNCHcast

Jon,

Attached is the information requested to get the LAUNCHcast patent moving forward.

Please let me know if you require additional information.

Regards,

peter.

From: From, Linda
Sent: Friday, October 29, 1999 3:43 PM
To: LAUNCHcast Developers
Subject: LAUNCHcast questions

I just noticed that on my DJs page and the Popular DJs pages, the checkboxes that indicate "subscribed" are not checked for the DJs to whom I have subscribed. Does this check indicate that a DJ has subscribed to you - or that you have subscribed to the DJ?

When I am on the LAUNCHcast DJs page and I click the "more" button at the bottom, I switch to the "Popular DJs" list. (BTW, the banner here says just "LAUNCHcast" - should it still say "LAUNCHcast DJs"?) On the Popular DJs pages, you can use the NEXT>> and <<PREVIOUS links to scroll back-and-forth through the list... My question is, why doesn't the <<PREVIOUS link take me all the way back to where I started from, my DJs page?

Thanks,

Linda

From: Sichel, Jon
Sent: Friday, October 29, 1999 5:37 PM
To: 'rauson@cisco.com'
Subject: FW: Inventor's Disclosure Short Form for LAUNCHcast

Here it is again

-----Original Message-----

From: Sichel, Jon
Sent: Thursday, October 28, 1999 8:44 PM
To: 'rauson@cisco.com'
Subject: FW: Inventor's Disclosure Short Form for LAUNCHcast

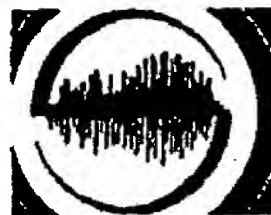
Bob - as promised, here are documents from our LAUNCHcast folks. After you've reviewed them, let me know the next steps and the projected costs for the retainer letter. Thanks for your help!

Jon

FAX COVER SHEET

LAUNCH

launch.com • Discover New Music



Date: 10-29-99
Attention: Bob Lawson
Company:
Fax: 310.394.4477
Tel:
From: Holly
Direct Tel: 310.526.4310

Patent Information
Patent Page

This fax consists of 29 page(s) including cover sheet.

Message: Bob,
Per Jon's request, attached please find
the patent information.
thanks,
Holly

Confidentiality Notice

This communication is ONLY for the person named above. Unless otherwise indicated, it contains information that is confidential, privileged or exempt from disclosure under applicable law. If you are not the person named above, or responsible for delivering it to that person, be aware that disclosure, copying, distribution or use of this communication is strictly PROHIBITED. If you have received it in error, or are uncertain as to its proper handling, please immediately notify us by telephone and return the original to us at the below noted address. Thank you.

2700 Pennsylvania Avenue Santa Monica CA 90404 Tel: 310.526.4300 Fax: 310.526.4400

From: Boulter, Jeff
Sent: Friday, October 29, 1999 6:45 PM
To: From, Linda
Cc: Beaupre, Todd
Subject: RE: LAUNCHcast questions

The first thing you mention is a bug. It's fixed on devweb6.

Interesting idea on the second thing. The previous and next buttons are meant for scrolling within a list on page. I guess the philosophy is to use the previous and next to page through a list, and use your browser's back button to go back to the previous page? We'll give it some more thorough thought.

Jeff

From: Boulter, Jeff
Sent: Sunday, October 31, 1999 3:40 PM
To: Leung, Ted
Subject: ratings cache reset?

Did you implement anything in the ratings cache that would reget all the ratings in the cache every X hours? I thought you did, but I could not find it in the code.

Something funky's happening and some ratings that are in the database are not in the ratings caches.

Jeff

From: Beaupre, Todd
Sent: Monday, November 01, 1999 9:30 AM
To: Fisher, Roberto; Glassley, Bob; Goetz, Richard; Bosick, Tom; Hughes, Jim
Cc: Boutier, Jeff; Woolsey, Julian
Subject: LAUNCHcast Perf Test

Any reason why we can't send out e-mails at 1 PM for a 3 - 4 PM LAUNCHcast performance test?

I will call Greg Messner to make sure we have all appropriate perf data collection in place.

From: Boulter, Jeff
Sent: Monday, November 01, 1999 2:40 PM
To: Beaupre, Todd
Subject: RE: LAUNCHcast Perf Test

use the same powerdog login

-----Original Message-----

From: Beaupre, Todd
Sent: Monday, November 01, 1999 10:58 AM
To: Beaupre, Todd; Fisher, Roberto; Glassley, Bob; Goetz, Richard; Bosick, Tom; Hughes, Jim
Cc: Boulter, Jeff; Woolsey, Julian
Subject: RE: LAUNCHcast Perf Test

We're all set to go from Greg Messner's point-of-view. I will assume the same from the rest of you, since I haven't heard anything to the contrary.

The only thing we really need to do is to add "# of connections" to the DATABASE 3 perf counter collection set, recycle the perf data log service there. Who has PCAnywhere access to DB3 to do this?

-----Original Message-----

From: Beaupre, Todd
Sent: Monday, November 01, 1999 9:30 AM
To: Fisher, Roberto; Glassley, Bob; Goetz, Richard; Bosick, Tom; Hughes, Jim
Cc: Boulter, Jeff; Woolsey, Julian
Subject: LAUNCHcast Perf Test

Any reason why we can't send out e-mails at 1 PM for a 3 - 4 PM LAUNCHcast performance test?

I will call Greg Messner to make sure we have all appropriate perf data collection in place.

From: Boulter, Jeff
Sent: Monday, November 01, 1999 6:35 PM
To: NetHelp
Subject: RE: lcplaylist1 and lcplaylist2

C:\>nslookup lcplaylist1.launch.com
Server: vns-c-pri.sys.gtei.net
Address: 4.2.2.1

*** vns-c-pri.sys.gtei.net can't find lcplaylist1.launch.com: Non-existent domain

-----Original Message-----

From: Zadjmool, Ray On Behalf Of NetHelp
Sent: Monday, November 01, 1999 6:33 PM
To: Boulter, Jeff
Subject: RE: lcplaylist1 and lcplaylist2

Done,
Please verify.

-----Original Message-----

From: Boulter, Jeff
Sent: Monday, November 01, 1999 6:22 PM
To: NetHelp
Cc: Beaupre, Todd
Subject: lcplaylist1 and lcplaylist2

Can you add DNS entries for these machines? I can hit their IPs, but I keep forgetting them.

Thanks,

Jeff

From: Boulter, Jeff
Sent: Monday, November 01, 1999 5:45 PM
To: Beaupre, Todd
Subject: RE: Legal rules bug

I was here all weekend fixing your ##### bugs

-----Original Message-----
From: Beaupre, Todd
Sent: Monday, November 01, 1999 2:44 PM
To: Boulter, Jeff
Subject: RE: Legal rules bug

Also, if you want me to work to stop nagging you, you have to try to do a better job convincing me that you feel the problems I bring up are important. Sometimes, you just shrug me off like "I don't know what the problem is. I can't reproduce it. Quit bringing it up." which only makes me want to push it more because I don't see you recognizing the important problems.

-----Original Message-----
From: Boulter, Jeff
Sent: Monday, November 01, 1999 2:33 PM
To: Beaupre, Todd
Subject: RE: Legal rules bug

again. This is not useful to me, and just comes off as nagging.

Jeff

From: Boulter, Jeff
Sent: Tuesday, November 02, 1999 8:18 PM
To: 'Glenn Thomas - Smashing Ideas'
Subject: RE: current player

Glenn, this player crashes IE either while it's loading or immediately before it starts playing music. I'm on NT 4 with IE 5.

Jeff

-----Original Message-----

From: Glenn Thomas - Smashing Ideas [mailto:glenn@smashingideas.com]
Sent: Tuesday, November 02, 1999 6:14 PM
To: Boulter, Jeff
Subject: current player

Jeff,

Here is the current player. I will forward on the list with responses, but most all of them should be fixed except the javascript related ones and the empty one with netscape.

I am going through trying to make it smaller now.

Glenn
Smashing Ideas
www.smashingideas.com
206.378.0100

From: Rho, Angie (mailto:Rho, Angie) On Behalf Of Rho, Angie

Sent: Wednesday, November 03, 1999 5:38 PM

To: Hughes, Jim; Beaupre, Todd; Boulter, Jeff; Goldberg, Dave; Feinberg, Catherine; Sichel, Jon; Gorla, Peter

Subject: Memo for DMCA Meeting

enclosed. Please take a quick look at the enclosed document for our meeting at 6:00 in Dave's conf. room. Thanks, Angie



<<...>> LAUNCHcast DMCA Memo.doc

From: From, Linda

Sent: Wednesday, November 03, 1999 4:50 PM
To: Bishop, Shane; Web Product Managers
Cc: Gorla, Peter
Subject: 5 best things about LAUNCH

Okay, here's your chance to plug your product! The redesigned registration form will include the "Top 5 Reasons to Join LAUNCH" - so please send me your ideas for the best things about LAUNCH! I was thinking maybe one each for watch, listen, read, interact, and win - but I'm open to other suggestions. If you could send me any ideas you have by 11AM tomorrow morning, that would be great!

Thanks for your help!

Linda

From: Boulter, Jeff
Sent: Wednesday, November 03, 1999 5:49 PM
To: NetHelp
Subject: RE: lcplaylist1 and lcplaylist2

That works, but why launchcast.com?

Jeff

-----Original Message-----

From: Zadjmool, Ray On Behalf Of NetHelp
Sent: Wednesday, November 03, 1999 2:39 PM
To: Boulter, Jeff
Subject: RE: lcplaylist1 and lcplaylist2

TRY THIS

LCPLAYLIST1.LAUNCHCAST.COM
LCPLAYLIST2.LAUNCHCAST.COM

From: Boulier, Jeff
Sent: Thursday, November 04, 1999 10:31 PM
To: 'ajordan@cisco.com'
Cc: Beaupre, Todd
Subject: RE: First Questions

Andrew,

changeRatee only performs one function only - it notifies the flash player that there's a new song being played. We used to pass all kinds of data to it, but for now we only pass 4 things:

clipID - an id for this particular media
clipTitle - the name for the clip, displayed when an ad, news, or tip plays
clipURL - a URL to link the title to when an ad, news, or tip plays
clipType - one of "ad", "news", "tip", "interstitial", "news", "broadcast", "song", or "video"

playerControl was intended to change the visual state of the player interface if something changed on the media player side. We never really used it before, but we are using it now to tell the player to indicate when media player is buffering a song before play.

In general, media player fires JavaScript events which we redirect to flash to update the state of the player. On occasion, we communicate the other way from flash to media player such as when a user clicks the pause button.

You can direct future questions of a more technical nature to me.

Jeff

-----Original Message-----

From: ajordan@cisco.com [mailto:ajordan@cisco.com]
Sent: Thursday, November 04, 1999 7:20 PM
To: Beaupre, Todd
Subject: First Questions

Todd,

I am beginning to go closely through the fax with the Interface and Playlist Generator and I would like to pose my questions to you, if you have time. I would like to use you as my contact person. It's OK to decline, but it's pretty important that my questions get answered, somehow.

The first one has to do with the variables in the flashplayer1.htm file with its description of the LAUNCHcast Player Interface.

If changeRatee does not use all the variables listed, what is the role for the variables? What uses them? Do changeRatee and playerControl provide complete control for the user over the player interface? Are there standard protocols of which I should be aware? It looks like the user interface and the music stream are handled separately, but in a coordinated fashion. Please confirm.

Andrew

From: Boulter, Jeff
Sent: Thursday, November 04, 1999 3:17 PM
To: Beaupre, Todd
Subject: RE: DJ sorting in player

make a bug

~~-----Original Message-----~~

From: Beaupre, Todd
Sent: Thursday, November 04, 1999 12:11 PM
To: Boulter, Jeff; Lee, Howard
Subject: DJ sorting in player

DJs sorted incorrectly. Used to sort by rating.

<< OLE Object: Device Independent Bitmap >>

From: Boulter, Jeff
Sent: Thursday, November 04, 1999 11:33 PM
To: Fritz, Cort
Subject: launchcast architecture, the long version



architecture.txt

Jeff Boulter
jeffb@launch.com
Senior Director, Product Development
Launch Media, Inc.
www.launch.com
310-526-4387

From: ajordan@cislo.com [mailto:ajordan@cislo.com]
Sent: Thursday, November 04, 1999 8:41 PM
To: Beaupre, Todd
Subject: Widgets

Is there a specific definition for a widget, or is it a generic term for any small piece of accompanying software?

Please contact me by telephone or reply email if you have any questions or comments.

Very Truly Yours,

Cislo & Thomas LLP

Andrew S. Jordan

From: Boulter, Jeff
Sent: Friday, November 05, 1999 3:58 PM
To: 'Glenn Thomas - Smashing Ideas'
Subject: RE: issues

Don't the indexes start at 0?

-----Original Message-----

From: Glenn Thomas - Smashing Ideas [mailto:glennt@smashingideas.com]
Sent: Friday, November 05, 1999 1:00 PM
To: Boulter, Jeff
Subject: Re: issues

also, has anything changed with the variable naming for radio_name? If I force a radio_name1 and radio_id1 through the Flash player here, it comes up but I can never seem to get anything to come up from the radios I've selected. I'll add more radio stations and keep trying.

Glenn
Smashing Ideas
www.smashingideas.com
206.378.0100

----- Original Message -----

From: Boulter, Jeff <jeffb@launch.com>
To: 'Glenn Thomas - Smashing Ideas' <glennt@smashingideas.com>
Sent: Friday, November 05, 1999 12:51 PM
Subject: RE: issues

> Try logging out and log back in again

>

> -----Original Message-----

> From: Glenn Thomas - Smashing Ideas [mailto:glennt@smashingideas.com]
> Sent: Friday, November 05, 1999 12:53 PM
> To: Boulter, Jeff
> Subject: Re: issues

>

> Has something been changed with the djName? I'm getting a Not+Available
> returned to mine and so is our other tester.

>

> Glenn
> Smashing Ideas
> www.smashingideas.com
> 206.378.0100

>

> ----- Original Message -----

> From: Boulter, Jeff <jeffb@launch.com>
> To: Glenn Thomas (E-mail) <glennt@smashingideas.com>
> Sent: Friday, November 05, 1999 10:25 AM
> Subject: issues

>

> > Not many left!

>

> > Flash Player issues

>

> > 29. player - I witnessed one user for whom the player would not load at

>

> > all
> > in Netscape. All it would do is remain white. Right clicking on the

>

> white

1.11.11.11.11

> > area showed that flash 4 was installed, but the movie was not loaded.
>> Update: I saw this happen once for the spinning logo too, so it seems to
be
> > a flash issue. I've contacted Macromedia.

> >
> >

PRIVILEGED MATERIAL
REDACTED

> >
> > 61. The player doesn't seem to be showing radio stations that play it.
> >
> > 57. player & horizontal widget - when you unrate something, it sends two
> > ratings out - one for the first click, and a second that unrates it
> >
> > 59. player - under what conditions is the updating movie playing between
> > songs? It seems to be fixed for ads, but I don't see it play between
songs
> > anymore too.
> >
> > 60. Can you make sure the color of the player looks OK in 16-bit color.
> The
> > web color I have for the gifs below the ad is #9898AC
> >
> > 55. player - is it me, or does the updating movie not "hurry up" anymore
> > once it gets the song info?
> >
> >
> >
> > Jeff Boulter
> > jeffb@launch.com
> > Senior Director, Product Development
> > Launch Media, Inc.
> > www.launch.com
> > 310-526-4367

From: Boulter, Jeff
Sent: Friday, November 05, 1999 8:09 PM
To: Beaupre, Todd
Subject: Rating/Personalizing



LAUNCHcast FAQ
redline v3_.doc...

~~-----Original Message-----~~

From: Hughes, Jim
Sent: Friday, November 05, 1999 12:08 PM
To: Beaupre, Todd; Boulter, Jeff
Cc: Goldberg, Dave; Mico, Ted; Gorla, Peter
Subject: latest lc player

I think the player would look much cleaner if it were
just the launch logo in bottom right; repeating
LAUNCHcast is a little overkill and really fights
w/animated ads...
my opinion
jim

PTO-103P
(Rev. 8-89)

**PROVISIONAL APPLICATION
FILING RECEIPT**



**UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office
ASSISTANT SECRETARY AND COMMISSIONER
OF PATENTS AND TRADEMARKS
Washington, D.C. 20231**

APPLICATION NUMBER	FILING DATE	GRP ART UNIT	FIL FEE REC'D	ATTORNEY DOCKET NO.	DRWGS	TOT CL	IND CL
60/164,846	11/10/99		\$150.00	D-7879	31		

ANDREW S JORDAN ESQ
CISLO & THOMAS LLP
233 WILSHIRE BOULEVARD SUITE 900
SANTA MONICA CA 90401-1211

Receipt is acknowledged of this Provisional Application. This Provisional Application will not be examined for patentability. Be sure to provide the PROVISIONAL APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION when inquiring about this application. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. If an error is noted on this Filing Receipt, please write to the Office of Initial Patent Examination's Customer Service Center. Please provide a copy of this Provisional Application Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts of Application" ("Missing Parts Notice") in this application, please submit any corrections to this Filing Receipt with your reply to the "Missing Parts Notice." When the PTO processes the reply to the "Missing Parts Notice," the PTO will generate another Filing Receipt incorporating the requested corrections (if appropriate). This Provisional Application will automatically be abandoned twelve (12) months after its filing date and will not be subject to revival to restore it to pending status beyond a date which is after twelve (12) months from its filing date.

Applicant(s) **JEFFREY R. BOULTER, LOS ANGELES, CA; TODD M. BEAUPRE,
LOS ANGELES, CA.**

IF REQUIRED, FOREIGN FILING LICENSE GRANTED 12/06/99
TITLE
INTERNET RADIO AND BROADCAST METHOD

DATA ENTRY BY: YON, LOWUAN

TEAM: 05 DATE: 12/06/99

(See reverse for new important information)



UNITED STATES PATENT AND TRADEMARK OFFICE

COMMISSIONER FOR PATENTS
UNITED STATES PATENT AND TRADEMARK OFFICE
Washington, D.C. 20231
www.uspto.gov

APPLICATION NUMBER	FILING DATE	GRP ART UNIT	FILE REC'D	ATTY. DOCKET NO.	DRAWINGS	TOT CLAIMS	IND CLAIMS
09/709,234	11/09/2000	2681	382	00-8832	3	23	3

CONFIRMATION NO. 3098

FILING RECEIPT



0000000005881041

Andrew S. Jordan, Esq.
Cislo & Thomas LLP
Suite 900
233 Wilshire Blvd.
Santa Monica, CA 90401-1211

Date Mailed: 03/14/2001

Receipt is acknowledged of this nonprovisional Patent Application. It will be considered in its order and you will be notified as to the results of the examination. Be sure to provide the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION when inquiring about this application. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. If an error is noted on this Filing Receipt, please write to the Office of Initial Patent Examination's Customer Service Center. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the PTO processes the reply to the Notice, the PTO will generate another Filing Receipt incorporating the requested corrections (if appropriate).

Applicant(s)

Jeffrey R. Boulter, Los Angeles, CA;
Todd M. Beaupre, Los Angeles, CA;

Continuing Data as Claimed by Applicant

THIS APPLN CLAIMS BENEFIT OF 60/164,846 11/10/1999

Foreign Applications

If Required, Foreign Filing License Granted 03/14/2001

Projected Publication Date:

Non-Publication Request: No

Early Publication Request: No

** SMALL ENTITY **

Title

Internet radio and broadcast method

Preliminary Class
455

Data entry by : WOLDEYES, TEGUEST

Team : OIPE

Date: 03/14/2001

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000

**LICENSE FOR FOREIGN FILING UNDER
Title 35, United States Code, Section 184
Title 37, Code of Federal Regulations, 5.11 & 5.15**

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 38 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Office of Export Administration, Department of Commerce (15 CFR 370.10 (j)); the Office of Foreign Assets Control, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15 (b).

PLEASE NOTE the following information about the Filing Receipt:

- The articles such as "a," "an" and "the" are not included as the first words in the title of an application. They are considered to be unnecessary to the understanding of the title.
- The words "new," "improved," "improvements in" or "relating to" are not included as first words in the title of an application because a patent application, by nature, is a new idea or improvement.
- The title may be truncated if it consists of more than 600 characters (letters and spaces combined).
- The docket number allows a maximum of 25 characters.
- If your application was submitted under 37 CFR 1.10, your filing date should be the "date in" found on the Express Mail label. If there is a discrepancy, you should submit a request for a corrected Filing Receipt along with a copy of the Express Mail label showing the "date in."
- The title is recorded in sentence case.

Any corrections that may need to be done to your Filing Receipt should be directed to:

Assistant Commissioner for Patents
Office of Initial Patent Examination
Customer Service Center
Washington, DC 20231

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☒ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.